Call-A.P.P.L.E.™

Volume VI, Number 1 January 1983 $2.50 ($3.00 Canada, £1.75 UK.)

A Call -A.P.P.L.E. TUTORIAL:
Beginners Guide Page 9

A IS FOR APPLE:
Password Protection Page 13

A Call - A.P.P.L.E. TECHNIQUE:
Bottoms Up! Page 17

A Call -A.P.P.L.E. TECHNOTE:
RND is Fatally Flawed Page 29

Apple PugetSound Program Library Exchange

THE LEADING EDGE IN PRINTERS

ONE GREAT LINE. ONE GREAT WARRANTY.

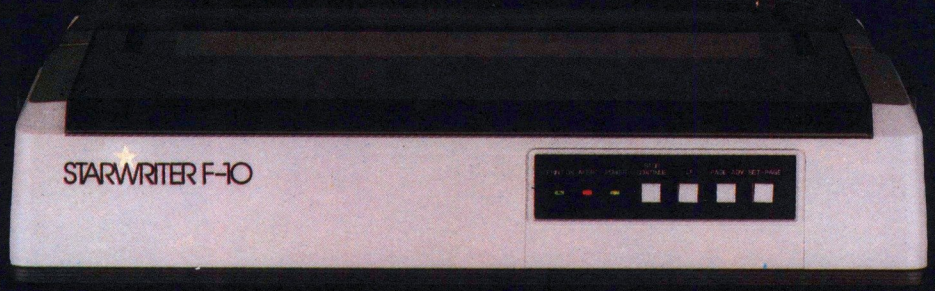
Finally, there's one full family of printers that covers every business or word processing application—all from C. Itoh, a company known for packing more product into less price; and all distributed exclusively by Leading Edge, a company known for searching out and providing that very thing. Which means that one call to one source can get you any printer, any time you need it, for any purpose. All backed by a full years' warranty from Leading Edge. (Try *that* on any other line of printers.)

THE PRO'S.

The Prowriters: business printers—and more. The "more" is a dot-matrix process with more dots. It gives you denser, correspondence quality copy (as opposed to business quality copy, which looks like a bad job of spray-painting).
Prowriter: 120 cps. 80 columns dot matrix compressable to 136. 10" carriage. Parallel or serial interface.
Prowriter 2: Same as Prowriter, except 15" carriage allows full 136 columns in normal print mode. Parallel or serial interface.

THE STAR.

The Starwriter F-10. In short (or more precisely, in a sleek 6" high, 30-pound unit), it gives you more of just about everything—except bulk and noise—than any other printer in its price range. It's a 40 cps letter-quality daisy-wheel with a bunch of built-in functions to simplify and speed up word processing. It plugs into almost any micro on the market, serial or parallel.



THE MASTER.

The Printmaster F-10. Does all the same good stuff as the Starwriter except, at 55 cps, the Master does it faster.

Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021. Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.

Vista DaVinci
GREAT INNOVATORS

In Italy in the 15th century Leonardo Da Vinci was a multi-talented genuis. Leonardo's ideas and designs were endless. He was always creating new devices to solve the world's problems. Pictured here are many studies for military arms and armor and battlements.

Today Vista is developing many high technology computer peripherals for use with IBM™ and Apple Computers. The final evolutionary stage in Apple disk storage, Quartet offers you the capacity of 4 Apple Disk Drives in the volume of one. Quartet's low profile styling has been developed to blend in with the profile and style of your Apple II Computer.

Quartet uses 2 Double Sided Thinline disk drives to give you over 640K of storage. With the included Quartet Controller & Software, your Computer can handle much larger tasks than before. When using protected software, the Quartet operates just like 2 Apple Disk Drives, and will operate with ALL Apple compatible software.

The Quartet system is the COMPLETE, one-stop solution to your storage requirements on the Apple or Apple Compatible Computer. No other two-drive package offers as much value at any price. And the Quartet offers all this with a price comparable to single drive, single sided systems.

Features: • Thinline Drives for low profile • 2 Double Sided Drives for 640K Storage • Quartet Controller for Single/Double Sided Operation • Emulates Apple Disk II under a single sided mode • Boots & Runs All Apple Software • Double Sided, 40 track Patch software for DOS, CP/M*, & Pascal • Full Vista 120 Day Warranty

Vista Quartet

Contact Your Local Vista Dealer or Call our Vista Hotlines.

Vista COMPUTER COMPANY, INC.

1317 East Edinger / Santa Ana, CA 92705
(714) 953-0523 / (800) 854-8017

DISTRIBUTORS / REPRESENTATIVES

Western - Group 3 Wholesale
(213) 973-7844 (408) 732-1307
South Central - M.P. Systems
(214) 385-8885
UK - Hal Computers Ltd.
(0252) 517175 / TWX: 858404
Northeast - Computers & Peripherals Inc.
(315) 476-6664
Florida - Audio Marketing (REP)
(305) 322-8327
Central - Wyatt & Associates
(317) 773-4791

*CP/M is a trademark of Digital Research, Inc.

TMIBM is a registered trademark of International Business Machines.

APPLE PUGETSOUND PROGRAM LIBRARY EXCHANGE:

CLUB INFORMATION

GENERAL OFFICES
304 Main Ave. S., Suite 300
Renton, WA 98055 — (206) 271-4514
Hours: 8 - 4 Mon. - Fri.
Member Services/Dealer Orders

EDITORIAL DEPARTMENT
5606 California Ave. S.W.
Seattle, WA 98136—(206)932-8288
Hours: 8-4 Monday - Friday

ADVERTISING REPRESENTATIVES
Grawin Publications
1020 Lloyd Building
Seattle, WA 98101 (206) 223-0861

MEETING NOTICES

7:30 PM, Tuesday, January 18, 1983

Red Lion Inn
18740 Pacific Hiway S.

Guest Speaker
Mike DeVille/Basis 108 Computer

7:30 PM, Tuesday, February 15, 1983

Bannan Auditorium
Seattle University

Speaker to be announced

7:30 PM, Tuesday, March 15, 1983

Speaker and location to be announced

CALL —A.P.P.L.E. HOT LINE
Technical/programming assistance only
(206) 932-6588
Hours: 9 AM - 3 PM and
6 PM - 10 PM 7 days per week
Pacific Time

Please refer Pascal, Apple ///, VisiCalc, questions directly to consultants on page 79.
See page 7 of this issue for information on use of the Hotline.
Please also check our list of consultants on page 79 for a specialist in your field.

Membership Information

Please direct telephone questions on membership and orders to (206) 271-4514.

One Time Apple—Cation Fee	$25.00
Annual Dues	$20.00
Total	$45.00

Additional postage required for foreign memberships and shipments.

Additional membership postage:

- Canada, West Indies, Central America, Columbia, Venezuela and Mexico — Add $15.00 ($60.00 total)
- All S. American countries (except above), Europe and N. Africa — Add $21.00 ($66.00 total)
- Asia, Mid East, S. Africa, Australia and Pacific Islands — Add $27.00 ($72.00 total)

All orders must be prepaid in U.S. funds only.

Membership is on an annual, 12 month basis, commencing with the month of enrollment. Upon receipt of total membership fees, new members will receive an Apple—Cation blank, order form and description sheet of club products.

Please allow up to 30 days for a turnaround on membership and orders.

Call —A.P.P.L.E. is mailed free twelve time per year as benefit of membership. Subscriptions are not available.

1982 back issues available at $2.50 each.

Call —A.P.P.L.E. bound volumes for previous years are available as indicated on the member order form on the last page.

Institutional memberships are now available. Contact General Office for details.

Prospective authors please send self-addressed stamped envelope (37¢ postage) to Editorial Dept. for a copy of Author's guidelines.

Cover and interior design by Vic Warren Design.

NOTICE

It is essential that membership number and expiration date be included on all orders and correspondence. This information is found on the top line of the mailing label of each issue of **Call —A.P.P.L.E.**

Please be aware of your expiration date so you can renew in time to avoid missing any issues of **Call —A.P.P.L.E.** Renewal notices are sent via first class mail approximately two months prior to the expiration date. Please check your own expiration date in the event your notice is not delivered.

Changes of address must be received at the General Office at least 30 days prior to effective date of move. Issues missed due to non-receipt of change of address notification must be separately reordered by the member at current cost. Change of address form may be found on page 79 of this issue.

A.P.P.L.E. CRATE BULLETIN BOARD
(206) 935-9119
(usually 24 hours)

Officers and Directors

Dick Hubert, President	(206) 271-6939
Bob Huelsdonk, Vice President	(206) 362-4910
Dave Lingwood, Secretary	(206) 244-9360
Bob Clardy, Treasurer	(206) 641-1917
Val J. Golding, Director	(206) 932-6588
Ken Smith, Director	(206) 322-4199
Fred Merchant, Director	(415) 921-7792

Office Staff

Dick Hubert	Executive Director
Dave O'Leary	Administrative Assistant
Sally Davies	Admin. of Information Systems
Arliss Sword	Dealer Orders
Patti Hoffmann	Special Projects Coordinator
Gunter Hirt	Bookkeeping, Member Records
Joff Morgan	Technical Operations
Cheryle McCalmant	Order Processing
Kris Mortenson	Order Processing
Sharon Birdsell	Order Processing
Gayle Rank	Secretary/Receptionist
Mike Christensen	Systems Development
Carol J. Mohn	Shipping/Purchasing
Brad Phillips	Shipping/Receiving
Steve Johnson	Ap·tronics
Steve Ross	New Product Development
Garry Reinhardt	Systems Development

Magazine Staff

Val J. Golding	Managing Editor	(206) 932-8288
Kathryn Hallgrimson	Editor	(206) 932-8288
Kathy Cherry	Editorial Assistant	(206) 932-8288
Jonathan V. Post	Contributing Editor	(206) 824-7641
Ron DeGroat	Pascal Editor	(303) 443-2722
Kevin O. Donohoe	Production Editor	(206) 223-0861
Jim Hockenhull	Staff Writer	(509) 334-4137
Clifton M. Howard, M.D.	Staff Writer	(201) 768-9274
Paul Mosher, M.D.	Staff Writer	(518) 765-4118
Bill Parker	Staff Writer	(619) 297-9343
Lee Reynolds	Staff Writer	(305) 722-2639
Clay Ruth	Staff Writer	(219) 865-6165
Charles Rusch	Staff Writer	(503) 484-4618
David G. Sparks	Staff Writer	(319) 232-2313
Ralph H. Swerdlow, M.D.	Staff Writer	(916) 920-8709
Grawin Publications	Production, Advertising	(206) 223-0861

A.P.P.L.E. consultants will be found on page 79 of this issue.

Call-A.P.P.L.E.™

Volume 6, Number 1 January 1983/Menu

Kathryn Hallgrimson Suther, Editor
Val J. Golding, Managing Editor

ENTRÉES

9 A CALL —A.P.P.L.E. TUTORIAL:
Beginner's Guide
Lee Webster

13 A IS FOR APPLE:
Password Protection
Ralph H. Swerdlow/Call —A.P.P.L.E. Staff Writer

17 A CALL —A.P.P.L.E. TECHNIQUE:
Bottoms Up!
Tom Little

29 A CALL —A.P.P.L.E. TECHNOTE:
RND is Fatally Flawed
David Sparks/Call —A.P.P.L.E. Staff Writer

À LA CARTE

33 **A New Pseudo—Random Number Generator**
John Russ

39 MACHINE POWER:
Where To Put Machine Code
Bill Parker/Call —A.P.P.L.E. Staff Writer

41 A CALL —A.P.P.L.E. PRODUCT REVIEW:
MX PLUS
Drew Berentes

43 **The Penultimate, Penultimate Input Anything Subroutine**
Bob Nacon

49 **Three Dee Demos**
David Sun

53 A CALL —A.P.P.L.E. TECHNIQUE:
Splitter
Norman L. Kushnick

57 **Direct to Page 2**
Bob Halfman

HORS d'OEUVRES

5 The Editor Bytes Back
The Editor

5 Boot
Craig MacIntosh

7 A.P.P.L.E. INFO:
A Live Wire
Val J. Golding

63 APPLEMASH:
Getting the Point
Michael Thyng

66 1982 Call —A.P.P.L.E. Index
Compiled by Kevin O. Donohoe and Val J. Golding

69 Write —A.P.P.L.E.
The Readers

77 Tomorrow's Apples Today
Kevin O. Donohoe/Editor

79 Advertiser's Index

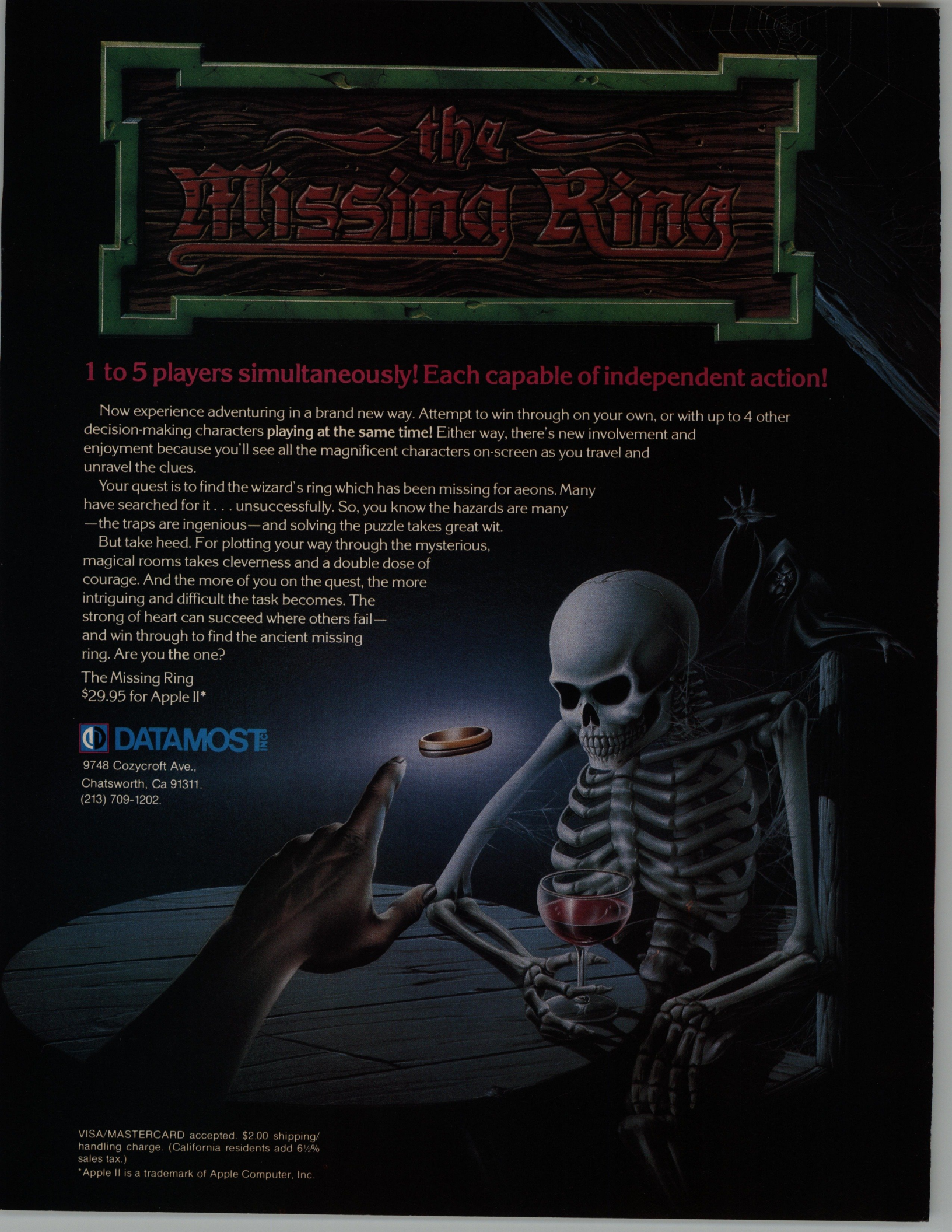
79 Call —A.P.P.L.E. Consultants

79 A.P.P.L.E. Order Form

80 Call —A.P.P.L.E. Unclassifieds

Apple user groups may reprint without prior permission any portion of the contents herein, provided author, title, and publication credits are given and a written notice of such use furnished to the editorial office. For commercial applications of programs herein, the author should be contacted c/o General Office. Call —A.P.P.L.E. is published monthly by Apple Puget Sound Program Library Exchange, 304 Main Ave. S., Renton, WA 98055. Subscriptions are not available but copies are mailed free as a benefit of A.P.P.L.E. membership.

Entire contents copyright© 1983 by
Apple PugetSound Program Library Exchange



1 to 5 players simultaneously! Each capable of independent action!

Now experience adventuring in a brand new way. Attempt to win through on your own, or with up to 4 other decision-making characters **playing at the same time!** Either way, there's new involvement and enjoyment because you'll see all the magnificent characters on-screen as you travel and unravel the clues.

Your quest is to find the wizard's ring which has been missing for aeons. Many have searched for it . . . unsuccessfully. So, you know the hazards are many —the traps are ingenious—and solving the puzzle takes great wit.

But take heed. For plotting your way through the mysterious, magical rooms takes cleverness and a double dose of courage. And the more of you on the quest, the more intriguing and difficult the task becomes. The strong of heart can succeed where others fail— and win through to find the ancient missing ring. Are you **the** one?

The Missing Ring
$29.95 for Apple II*

DATAMOST INC
9748 Cozycroft Ave.,
Chatsworth, Ca 91311.
(213) 709-1202.

VISA/MASTERCARD accepted. $2.00 shipping/ handling charge. (California residents add 6½% sales tax.)

*Apple II is a trademark of Apple Computer, Inc.

THE EDITOR BYTES BACK

the editor

A new year is upon us, another 12 issues of **Call -A.P.P.L.E.** We would like to request again our readers input. What are your preferences and dislikes. Take time, if the thought appeals to you, to photocopy the contents page and rate the stories. We're interested also in how you use your Apple, whether or not you write your own programs or plan to. Comments should be addressed to the Editorial Office at the address shown on page 2, and should include your membership number if a member.

So here we are with the brand new year. Perhaps your Christmas present to yourself was a shiny new Apple. If so, you can skip the rest of this page and turn right now to **Lee Webster's A Beginner's Guide,** which may prove just the ticket to get you off the ground with your new baby.

Step up one rung and take a look at Staff Writer **Ralph Swerdlow's** contribution to your peace of mind as he shows you how to implement **Password Protection** in your programs.

Another level up, **Tom Little** returns with **Bottoms Up,** some tips and techniques on structured programming, using a command intrepreter as an example.

At the bottom of the cover, but deepest into Apple's inner workings is Staff Writer **David Sparks' RND is Fatally Flawed,** a grand finale exposition of the problems with Applesoft's random number generator, for once and for all, we hope, laying the problem to rest. Dave here collaborates with **John Russ,** whose story **A New Pseudo-Random Number Generator** shares a program in common with Sparks' feature.

Bill Parker, who joins us this issue as a **Call -A.P.P.L.E.** Staff Writer, offers **Machine Power: Where to Put Machine Code,** one of a series of articles we plan for our two magazines.

Meanwhile, back on the contents page, we find one more version of the infamous "IAS," **Bob Nacon's Penultimate Penultimate Input Anything Subroutine** to once more let you enter commas, etc., in INPUTs, and we'd also like to call your attention to our annual **Call -A.P.P.L.E. Index for 1982,** compiled by coworker/cohort **Kevin Donohoe.**

Finally, to answer a often-asked Hotline question, **Bob Halfman** makes writing to text page 2 almost as easy as writing to page 1 with **Direct To Page 2.** Now with the appropriate POKEs, you can also use hi-res page 2 in mixed mode.

Next month, watch for a Valentine gift from **Apple Computer, Inc.,** as we lift the shroud and unveil details of the newest version of the Apple II computer, presented by Staff Writers **Cliff Howard** and **Art Schumer.** You will also find a new utility by **Ken Kashmarek** which we will wish to promote as a standard, a two byte PEEK/POKE routine which makes those 16 bit addresses even easier to handle than Integer Basic does.

A Call for Papers

Call it a half-baked idea . . . maybe, maybe not, but our ear has been bent by **Bill Parker** of San Diego who proposes that our club consider writing and releasing a new BASIC language. We would hope that noted authors such as **Cornelis Bongers, Dave Lingwood, C. K. Mesztenyi, Craig Peterson** and others could be coerced to participate. Through this column we are inviting those individuals to contact us, and we will work with **Bill Parker** as coordinators. If you are technically skilled in assembly and interested in the proposed project, let us know.

BOOT

Craig MacIntosh

Crown Syndicate, Inc., Five Crown Road, Weatherford, Texas 76086. Telephone (817) 594-5125.

EVERYONE NEEDS A . . .

UNINTERRUPTABLE POWER SOURCE

ADVANCED DESIGN
NEW RELIABILITY
GUARDIAN PROTECTS
ELECTRONIC SAFETY
LIFESAVER FOR DATA

$595

SAVE YOUR DATA FROM POWER OUTAGES!

BACKUP FOR YOUR COMPUTER, MONITOR, PRINTER AND 5¼" FLOPPY AND HARD DISC DRIVE

• Automatically stops annoying problems from power line interruptions and brown outs • You need standby power to save data • Maintenance free backup power available in 115 volt or 220 volt • 50 or 60 HZ • 150 watts • Complete versatility — operate your system from a 12 volt source, i.e., automobile cigarette lighter, boat or airplane • Rugged self contained gel cell battery • No voiding warranty — no cutting wires • Automatic audio alarm warning tone during commercial power failure or interrupt • UL listed • FCC approved • Transient voltage suppressor gives added insurance from line voltage spikes, utilizing Zener Ray™ • Green/red LED power status indicator • Green — normal AC line power • Slow blinking red — at least 6 minutes of remaining standby power • Fast blinking red — approximately 2 minutes of remaining battery power • Solid state technology unexcelled by any UPS power unit in its class.

RH ELECTRONICS, INC.

COPYRIGHT © 1981 - PATENTS PENDING

566 IRELAN, BUELLTON, CA 93427

(805) 688-2047

SEE YOUR RH ELECTRONICS PRODUCTS DEALER

FOR YOUR APPLE II*:

SUPER FAN II™	$ 74.95
SUPER FAN II™/ZENER RAY™	$109.00
SUPER RAM II™	$125.00
RH 12 VOLT TRANSVERTER	$149.00

FOR MICRO COMPUTERS:

GUARDIAN ANGEL™	$595.00

A.P.P.L.E. INFO:

A LIVE WIRE

Val J. Golding

SOME months ago we wrote on the proper use of the Call -A.P.P.L.E. Hot Line, and as our membership continues its rapid growth, we thought this might be an appropriate time to acquaint our new members with this service, and for the information of all, describe its beginnings.

First mention of the Hot Line appeared in the July, 1978, Call -A.P.P.L.E., which listed the newly-installed number, and briefly described the service, which was (and remains today) located in the basement of our (the editor's) home. Interestingly, the goals we set with that small announcement hold just as true today: to assist members with questions [answering programming problems], or referring them to some one [who specialized in their field of inquiry], who might be able to reply. Starting with the September, 1978 Call -A.P.P.L.E., the front cover listed telephone numbers for our consultants, a practice to be followed by some of the many—just now emerging—user groups.

In early 1980, when Call -A.P.P.L.E. became our full-time occupation, Hot Line hours were more than doubled to offer daytime service. Today the Hot Line handles upwards of 6000 calls from members and individuals each year. This could loosely be interpreted as saying about five out of eight members have availed themselves of the Hot Line during the last year.

As our rolls grow, so do our roles. The Hot Line becomes just a little bit busier each day. This in turn gives us the opportunity to tell you a bit about it, while at the same time how show you can help improve its efficiency.

Hot Tips

Hot line hours are from 9 AM to 3 PM and 6 PM to 10 PM, seven days per week. These hours were established only after considerable thought, and provide the maximum possible service hours to members in all time zones, while also allowing an uninterupted work period for ourselves. Our normal course of business requires we occasionally leave the telephone for brief periods, attending meetings or such mundane things as an emergency run to buy cat food. In these instances, a message recorder is normally on the telephone line. If you reach the recorder, please leave your telephone number first, including area code, your name, membership number, and the subject of the call. Knowing the subject can sometimes help us look up information before returning the call.

(continued on page 65)

500% Faster Loading of Textfiles, Integer, Applesoft & Binary and Now Six Powerful Extra Commands in Standard DOS 3.3

30 out of 60 bought it on the spot when DAVID-DOS™ was demonstrated at the Original Apple Corps, UCLA.

Automatically supports an Integer or Applesoft Rom Card in any slot (without configuring).

Six New Commands

1. TLOAD Speed loads all Text Files.
2. TLIST Lists all Text Files to screen/printer.
3. DUMP Binary/Ascii to screen or printer.
4. DISA Disassembles Binary to screen/printer.
5. AL Prints program Address & Length.
6. HIDOS DOS moves itself to Language Card

Like Other DOS Commands

These six commands operate identical to existing DOS commands. Use A or A$ for address and L or L$ for length. Enter them on the Keyboard and use them in Basic programs with the familiar D$.

100 Sectors in 7 Seconds

TLOAD loads a Visacalc Template of a hundred sectors in 7 seconds and TLIST scrolls it to screen or printer. TLOAD speed loads Sequential and Random Text Files starting at $1000 or as specified by A. It loads the entire file or a smaller length specified by L.

List Any Textfile

TLIST lists Sequential and Random Text Files to screen and printer.

DUMP screen dumps memory with Ascii along the right side. Use A and L for specific dumps. Without them it starts at $00 and stops with return.

Disassemble Machine Code

DISA disassembles machine code to screen. Use A and L for specific routines.

Variable Speed Scrolling

TLIST, DUMP and DISA Features:

1. Print with PR#
2. Key operated variable speed control.
3. Instant pause with Space Bar.
4. Exit anytime with Return Key.

10K More of Memory

HIDOS command checks for a language card in slot O and moves DOS residing in memory to the card at $DD00. Himem is set at $BE00 adding 10K of free memory for programs. The Integer Card is still automatically supported in any slot. $300 to $3D0 is kept free.

Apple II and Applesoft are trademarks of Apple Computer inc.
Visicalc is a trademark of Visicorp.

Two More New Commands

HIDOS has all previous commands and features operational and adds a 7th command FIND, Find searches 64K memory in 3 seconds for any Hex sequence up to 31 bytes long and prints the addresses to screen. Hidos has an optional 8th command DATE configured for Mountain Hardware clocks and is useable in basic programs.

Use with Most Software

DAVID-DOS is totally accurate and speed loads without over-writing. Great care was taken to keep DOS standard and compatible with most software. All DOS entry addresses have been preserved. Only two routines were moved to allow more room in command tables. DOS is the same length. Init areas were used for DAVID-DOS. Two extra variables are used in Zero Page.

Install in Seconds

Catalog Free Space is also provided. DAVID-DOS installs in seconds. Requires a 48K Apple II with DOS 3.3. Comes with instructions.

California residents add 6½% sales tax. **$39.95** Foreign add $2.00.

DAVID DATA™

To Order: Send Check To
12021 WILSHIRE BLVD., SUITE 212A
LOS ANGELES, CA 90025 (213) 478-7865

HERE ARE A FEW OF THIS MONTH'S SPECIALS

	Manufactures List Price	DELTREX Price
DELTREX LOWER CASE chip	$ 50.00	$ 9.50
M & R SUPERMOD][modulator	34.95	19.98
KENSINGTON SYSTEM SAVER fan	89.95	63.37
D C HAYES SMARTMODEM 300	289.00	199.00
D C HAYES MICROMODEM][	379.00	259.00
NOVATION APPLE CAT][modem	389.00	269.00
USI 9" HI-RES green monitor	149.00	109.00

	Manufactures List Price	DELTREX Price
BASF DISKS 10/box	$ 40.00	$ 24.98
DANA fan for APPLE	49.95	29.97
T G APPLE joystick	59.95	34.99
VIDEX ENHANCER	149.00	99.98
TRENDCOM 100 printer	295.00	229.89
RANA ELETE 1 drive	445.00	329.50
OKIDATA 82A printer	649.00	429.00

IF YOU ARE PAYING LIST PRICE, THEN YOU ARE PAYING UP TO 70% TOO MUCH.

DELTREX™ 2265 Westwood Blvd. Dept. #575 Los Angeles, CA 90064 (213) 558-1100

Orders $1,000 and below add 3% shipping. ($3.00 min.) CA add 6% (L.A. County 6½%). COD $2.00 extra

SHOP THROUGH OUR ENORMOUS SELECTION USING OUR 24 HOUR MODEM. (213) 202-6886

A Call —A.P.P.L.E. TUTORIAL:

Beginner's Guide

Lee A. Webster

Introduction

COMPUTERS need not be frightening!! They are simple high-speed idiots!! They need instructions to operate, and you can learn to "instruct" them, and use them, as well as anyone. Let's get you started in a relatively painless way. (I hope!)

To begin, locate an Apple II Plus computer (at a store, a friend's, a school or college), sit down, get comfortable, and take this article in hand. Let's begin.

Parts of the Computer System

Now that you are ready, look at each of the components (parts) on the table before you. They probably include:

(1) A CRT or Monitor, which looks like an ordinary television set—except it is used for computer output (display). In some cases, it may actually be a TV which has been adapted for computer use.

(2) A keyboard unit, which looks like a portable typewriter in a beige case. This is actually the Apple Computer and contains the computer chip (the brains of the outfit), memory, a power supply, and a bunch of other parts you don't need to know about yet.

(3) A small box with a slot in the front. This is the disk drive, which you'll use as a storage device (a way to save the work we do).

(4) Somewhere nearby there will also be a printer—it's the part with the paper hanging out of it. If there is no printer—don't worry, it won't be needed for this discussion.

Diskettes and Their Care

Take a deep breath now, and we'll begin your first encounter with the Apple. To begin, you need a floppy diskette. (Hopefully someone nearby has one to loan you.) A diskette looks like a 45 rpm record, with a large hole in the middle, except it is flexible. *Warning !!!* (don't mean to scare you already). You'll see an oval cutout and an area around the center of the diskette which is brown. This is the magnetic surface and you want to take care *not to touch* this surface. Hold the diskette only by the black case. Note also the surface of the diskette that actually *stores* the data is the reverse, or *unlabeled* side. Also, *do not bend* the diskette.

Bending or touching the surface can ruin the diskette so it can't be used again—and this could be disastrous if you have something important stored on it. You also want to *avoid* placing anything *on top* of a diskette when it is lying on a table, *avoid* writing on its label except lightly with a *felt* tip pen, *avoid* direct sunlight and hot/cold extremes, and *always* keep diskettes in their jackets to keep the dust off when they are not being used. (Note that the term "disk" is often used for "diskette".)

Hope you haven't been scared away, because other than the above warnings there is really nothing to worry about in using the computer. It is a very durable, and forgiving machine which will behave itself as long as you use reasonable care. Your typing or programming can't really damage the machine, so don't worry.

Beginning a Session on the Apple

You begin by turning both the Apple and the CRT *on*. On the CRT (or TV) find the ON/OFF switch/knob and turn it *on*. (If you are using a TV, you can also turn the volume *off* since you won't need it.)

Next, you need to insert a diskette into the disk drive (the box with the slot on the front). Open the disk drive door (if it isn't already open) by pulling outward on the bottom edge of the drive door (the vertical "tab" on the front of the drive). Take the diskette (your own or one borrowed from someone who knows what they're doing!!). Hold it in your right hand, with your thumb over the label. Now gently insert the diskette in the drive slot with the label facing *up* and the oval cutout toward the *back* of the drive. Push gently until it is entirely in, and close the drive door by pushing it down. The two metal fingers (which you'll see inside the slot when the door is closed) should just clear the diskette as the door is closed. If not, open the door and try again.

Time now to turn the computer *on*. The switch is on the *back* of the computer next to where the power cord plugs in. When you push this switch, the *power* light on the keyboard will light up (it is not a key). You will also note that the red light on the disk drive is on for a moment and there is a whirring sound from the drive unit—nothing to worry about. (If you forgot to insert a diskette in the drive, the drive will whirr forever—to stop it press the *reset* key on the keyboard—upper right corner of keyboard.) Some messages will appear on the CRT screen, then a] symbol and a flashing white square.

Doing Simple Calculations

Would you believe it?—you are now ready to "instruct" the Apple—tell it what *you* want it to do!! Let's assume you know nothing at this point (nothing personal, mind you!!). One thing you can do is to use the computer as a calculator. You can do this by knowing that the computer "speaks" and understands in a language called BASIC language. In that language:

—•→

+ means addition

- means subtraction

* means multiplication

/ means division

^ means exponentiation (powers)

Also, the computer won't talk back to you (i.e. display any answers on the CRT) unless you tell it to by using a PRINT statement.

To have the Apple give us the sum of 5 and 25, type:

```
PRINT  5+25
```

then press the RETURN key. Try it. (The answer, shown on the next line on the screen, should be 30)

A couple of words of explanation are in order here. The] sign is called a "prompt" which tells you that the Apple is presently working in Applesoft, a version of the BASIC language. The flashing white box is called the cursor and indicates the position on the screen where you are typing. The line you typed tells the computer to print (on the screen) the sum of 5 and 25. You can do any calculations this way, in what is called the "immediate execution mode." Remember—before the computer will do anything, it must receive your BASIC language instruction (in this case, the PRINT statement). It receives this line when you press the RETURN key. The RETURN key is almost always used to tell the computer we are through typing.

Correcting Typing Mistakes

If you make a mistake in typing (we all do!), it can be corrected easily if you haven't pressed RETURN yet. To correct, press the left-facing arrow key (right side of keyboard). Each time you press this key, the cursor moves back over the previous character. You can then re-type the characters. If you make a typing error and don't catch it before pressing RETURN, there is nothing to fear. The computer may beep and display an error message but no damage is done. The computer is just telling you it doesn't understand your typo! Just re-type the line. Now, try some other calculations. For example:

```
PRINT  345+345
PRINT  500-200
PRINT  3*6
PRINT  10/2
PRINT  3^2
```

By looking at the answers, can you tell what the last three PRINT statements instruct the computer to do? They are calculating products (multiplication), quotients (division), and powers (exponentiation). "3^2" is BASIC for 3-squared (3 x 3).

For what it's worth, you can now do calculations of mathematical expressions using the PRINT statement, numbers, and the +,-,*,/ and ^ symbols. You can also use parentheses. For example:

```
PRINT  (3+4)*(4+8)
```

calculates the product of two sums.

Running Programs Written by Others

I know, you are asking yourself why bother with this machine if all it will do is act like a $10 pocket calculator! There are other uses—many, many, many others. Most involve either using "canned" or pre-written programs or writing programs yourself. A *program* is nothing more than a sequence or list of instructions (like a recipe) telling the computer what to do and in what order. You can learn to program later, but for now you can use some "canned" programs (programs written by others).

See if you can borrow a diskette that contains some programs. Insert this disk in the disk drive (if you haven't already) by opening the door, removing the current diskette, and inserting the new one as instructed above.

Some System Commands

To find out what programs are contained on the disk, you use a SYSTEM COMMAND. The command is CATALOG. When you type this, and press the RETURN key, the disk will whirr in the drive and a list will appear on the screen. The list contains program names plus some other data you don't need to understand yet.

You may now have the computer follow the instructions (or "execute") any of the programs you see on the catalog list that are preceded by an A (for Applesoft). To do this, type:

```
RUN  name
```

Where "name" is one of the names on the catalog list. (Don't include the asterisk, single letter or three-digit number to the left of the title.) Be sure to spell it exactly as shown, complete with spaces and punctuation if present. Then press RETURN. The program should then execute. Follow the subsequent instructions in the program, remembering to press RETURN after typing in any response (s) it asks for.

When you are tired of working with one program, you may try another. If you want to stop a current program, press RESET (located at the upper right of the keyboard). You can now RUN another program, but first it is a good idea to type:

```
NEW
```

and press RETURN. This clears the computer's memory so it is ready for you to RUN another program.

The other system command you want to know is the LIST command, which causes the lines of the program to be displayed on the screen. You might want to do this with a program after you have RUN it, just to see what it looks like. The lines will soon roll past on the screen—just before you go crazy trying to read them, press the CTRL key (left side of keyboard) and *at the same time* press the C key. This will halt the listing. (This rolling of lines, incidently, is called scrolling).

What Happens If???

What If??? What if something doesn't happen exactly as explained above? Your best bet is to turn the Apple *off* by the switch on the back, then turn it *on* again, and start over (with a disk in the drive). There are other ways of handling this, but for now, this is good enough—and it always works!!!

When You Are Finished

When you have finished your session on the Apple, do the following:

(1) Remove the diskette

(2) Turn *off* the Apple by the switch on the back (the POWER light should go off)
(3) Turn *off* the CRT (or TV) using the ON/OFF switch.

Conclusion

At this point, you know (hopefully) how to get started on the Apple, how to do calculations (in what is called "immediate mode") and how to run "canned" or pre-written programs. If you now want to move on to bigger and better things, like writing your own programs, we suggest that you try one of the references listed, or look for a course at a local college or computer store or computer club.

We hope that this first experience hasn't been too harrowing. Please don't be intimidated by the computer—remember, you are smarter than it is!!! Just relax and enjoy. Computers are fun. Good Luck!!!!!

References You Should Know About

1. If you are a novice, the "Applesoft II Tutorial Manual" will provide a step-by-step guide to using the Apple, the disk system, and the extended BASIC computer language.
2. If you already know BASIC or want a refresher after using Reference 1, consult the "Applesoft II BASIC Programming Reference Manual." This contains the details of the extended Basic Language, in a reference format.
3. If you are experienced in BASIC and want to know how to handle the details of random vs. sequential data files, chaining programs, disk operating system (DOS), etc., consult the "Apple DOS Version 3.3 Instructional and Reference Manual."
4. If you are interested in the hardware details, peripheral interfacing machine or assembly language, memory mapping, etc., consult the "Apple II Reference Manual."

(Another alternative is the "Apple II User's Guide," by Poole, McNiff, and Cook, Osborne-McGraw/Hill, Berkeley, Ca., which contains material similar to that in references 2, 3, and 4 above. This is available to A.P.P.L.E. members at $13.00

WHAT ARE YOU WAITING FOR???

Are you tired of waiting for DOS to load and save files? Are you tired of waiting for DOS to finish so you can type again? Are you tired of waiting for your printer? When you buy **Diversi-DOS**™, you won't have to wait any more! Here's why:

1. DOS speed-up: Apple DOS 3.3 takes 18 disk revolutions to read a single track, whereas **Diversi-DOS** reads or writes a track in just 2 revolutions. This speeds up file processing tremendously (see table).

2. Keyboard Buffer: **Diversi-DOS** allows you to type at any time, as fast as you can, without missing a single character.

3. Print Buffer: **Diversi-DOS** can use a RAM card (16K-128K) to temporarily save characters before they are printed. Thus, your computer won't have to wait for your printer to finish.

Diversi-DOS, the TRIPLE utility, requires a 48K Apple II or II+ with DOS 3.3. A simple, menu-driven installation program is included on the un-protected disk. So what are you waiting for?

Send $30 to:

Diversified Software Research, Inc.
5848 Crampton Ct.
Rockford, IL 61111
(815) 877-1343

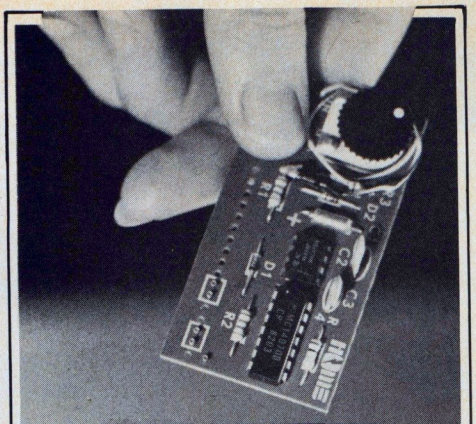
Visa/Mastercard accepted

	APPLE DOS	DIVERSI-DOS
SAVE ‡	27.1 sec.	5.9 sec.
LOAD ‡	19.2 sec.	4.5 sec.
BSAVE*	13.6 sec.	4.1 sec.
BLOAD*	9.5 sec.	2.6 sec.
READ**	42.2 sec.	12.4 sec.
WRITE**	44.6 sec.	14.9 sec.

* Hi-res screen ‡ 80-sector BASIC program
** 52-sector random access text file

Apple is a registered TM of Apple Computer, Inc.

dsr



SPEED UP
WORD PROCESSING
PROGRAM EDITING
VISICALC

the REPEATERRRR™
WITH
AUTO REPEAT & HIGH SPEED CURSOR

"The best Apple II keyboard enhancement for the money."
"Excellent documentation, very easy to install."
"LLLLove your Repeaterrrr!"

Apple II keyboard adapter

Repeats any keypress, including control characters for rubout, scrolling, etc. Featured on professional terminals. On/off control and adjustable delay to match your typing touch.

high-speed repeat

TWO SPEED REPEAT, keyboard selectable. The best part. Touch Apple's repeat key to DOUBLE the repeat rate. Great for long cursor moves. Zip through text, program, or across a VisiCalc spreadsheet to get there **FAST.** A must for 80 column displays.

easy installation

Remove Apple top and plug onto 25-pin connector between keyboard and encoder board (Rev. 7 or later). Compatible with all software, hardware, and Apple warranty.

no risk—30-day trial

Check, MasterCard, VISA (incl. card no. and exp. date). Prompt shipments. Full refund within 30 days.

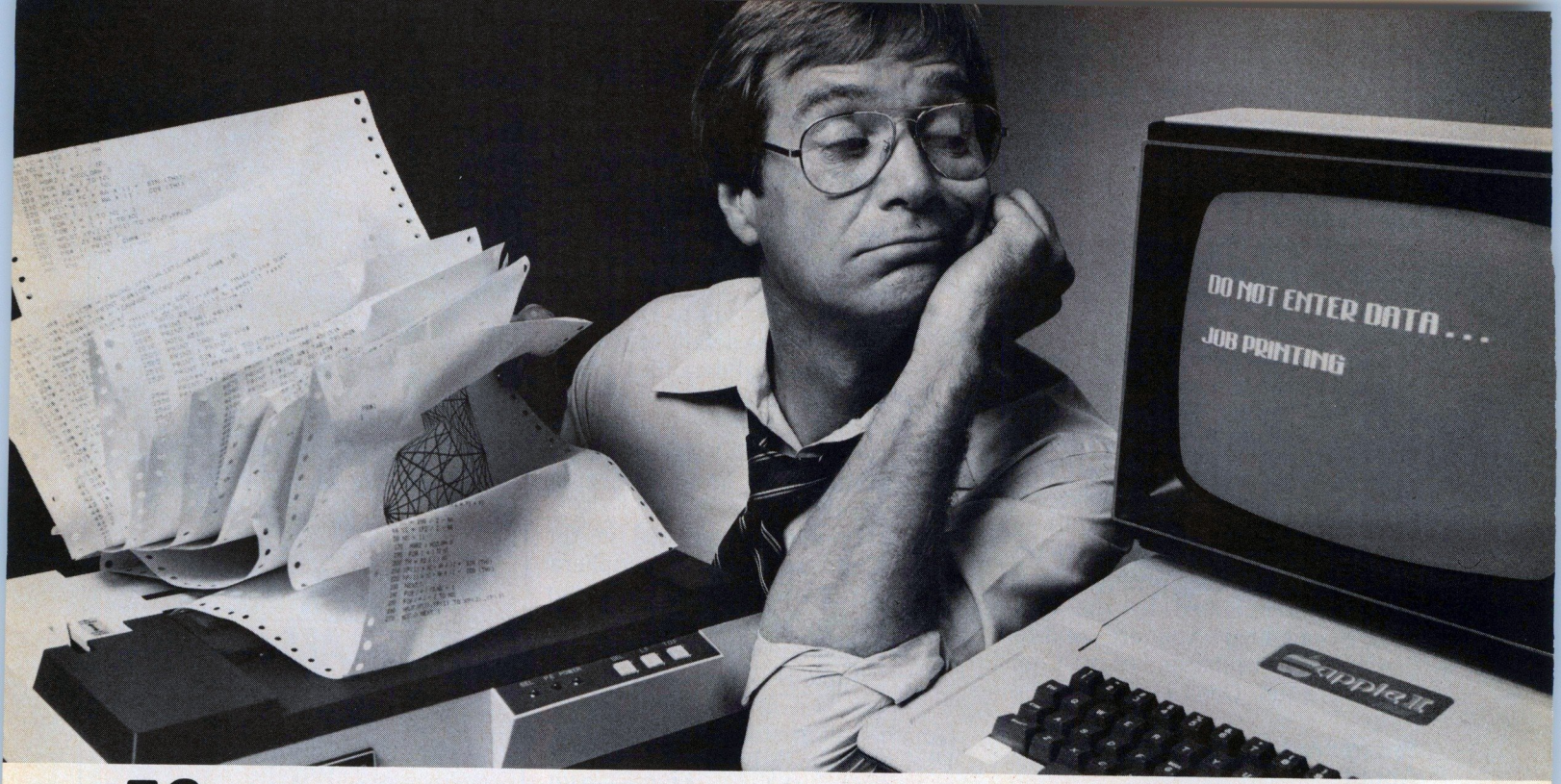
only $24.95 each

Add $2.00 per order shipping/handling ($5.00 foreign). Ohio orders add $1.63 per unit sales tax.
Dealer inquiries invited.

HOME

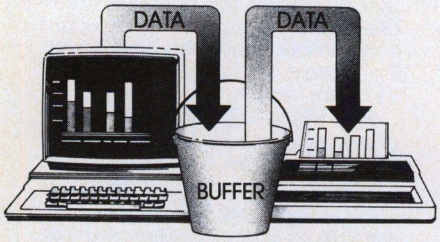
HIGH ORDER MICRO ELECTRONICS
17 RIVER ST. CHAGRIN FALLS OH 44022
PHONE 216-247-3110

APPLE IS A TRADEMARK OF APPLE COMPUTER, INC.
VISICALC IS A TRADEMARK OF VISICORP.



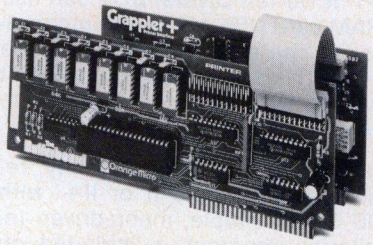
If your printer uses your Apple® more than you do, you need The Bufferboard.™

If your Apple is locked into the "PRINT" mode so much that you've taken up solitaire to kill the boredom, you need a buffer. And if your computer is the Apple II or III, the only buffer for you is The Bufferboard. Expandable to 64K of storage, The Bufferboard stores an instantaneous **bucketful** of print data from your computer. Then it feeds the data to your printer at its own printing rate. Your Apple is set free from driving your printer and is ready for more data from you.



Take your existing interface—and buffer it!

Only The Bufferboard has a simple Interface-Docking System. No bulky boxes or expensive power supplies are needed because The Bufferboard fits right into your Apple—and **docks** onto your existing printer interface. The result is convenient



and economical buffering of most popular printer interfaces, including the Grappler + ™ interface, Epson interface, and Apple printer interface. Thirty seconds and a single hook-up are all you need to end the printer waiting game forever.

Up to 20 letter-size pages stored at a time.

The Bufferboard comes standard with 16K, and is expandable to 32K or 64K of buffering capacity with the addition of memory chips. This "bucket" will hold up to 20 pages of a print job, allowing you freedom to use your Apple.

The Bufferboard—designed exclusively for the Apple Computer.

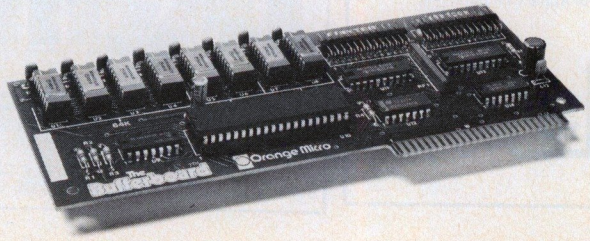
Specifications:

• Versions for Grappler + interface, Epson interface, Apple interface, and other popular printer interfaces • 16K buffer standard • Upgradeable to 32K or 64K • Automatic memory configuration • Automatic self test • Includes interface docking cable.

The Bufferboard is made by Orange Micro, Inc.; the same people who brought you the popular Grappler + printer interface. Both the Grappler + and The Bufferboard are now available at your local Apple dealer.

Apple is a registered trademark of Apple, Inc.
Epson is a registered trademark of Epson America, Inc.

Orange Micro inc.
1400 N. Lakeview, Anaheim, CA 92807
U.S.A. (714) 779-2772
TELEX: TX 183511 CSMA



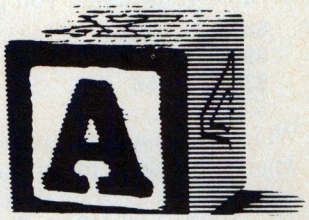
The Bufferboard™
For Apples and Printers

© Orange Micro, Inc. 1982

A IS FOR APPLE:

Password Protection

Ralph H. Swerdlow, M.D./Call —A.P.P.L.E. Staff Writer



HAVE you ever had a program or a disk that contained some material that you considered confidential? Such records might consist of financial information, business contacts, or research data that you would prefer to remain inaccessible to others.

This can be done in many ways, for example, by modifying DOS or using some sort of code to record and recover the information. This requires either a great deal of programming skill or an extra step or two to be able to protect this sensitive information. An easier way to accomplish the same thing would be to use a password protection scheme that will enable you to easily use the disk and program, but will prevent the unauthorized reading or changing of the data on the disk unless the correct password is used.

There are two ways to do this. The first is to use the password program as the Hello or greeting program, which would prevent the disk from being used, or as the first few lines of a separate program. If the latter is chosen, the disk could be used normally, but the specific program could not!

It must be obvious that the only totally effective way to protect your files is to hide or to lock away the disk that contains them, and that short of this, there is no foolproof method. A password will only slow down an experienced Apple user, but this may be more than enough to protect your files. I will not go into any details as to circumventing the use of a password. Hopefully, the use of this protection will be more than sufficient for your needs. Even if it is not, studying this program will teach you something about your Apple. Let us look at the program.

```
]LIST

10   REM  PASSWORD 9/14/82
20   REM  NORMAL 214=0
       1010=191 1011=157
       1012=56
30   TEXT
40   POKE 214,128
50   POKE 1010,102
60   POKE 1011,213: POKE
       1012,112
70   HOME
80   VTAB 12
90   INPUT "PLEASE ENTER
       THE PASSWORD :";PW$
100  REM  PASSWORD GOES
       HERE
110  IF PW$ = "GLORIA"
       THEN 140
120  HOME : FLASH :
       VTAB 12: HTAB
       6: PRINT "SORRY,
       YOU DON'T KNOW
       THE PASSWORD":
       NORMAL : PRINT
       : PRINT  CHR$
       (4);"PR#6"
130  END
140  HOME
150  HTAB 8
160  VTAB 10
170  HTAB 5: PRINT "THAT
       IS THE CORRECT
       PASSWORD"
180  PRINT : PRINT
       TAB( 10)"LET'S
       GET TO WORK."
190  REM  POKE 214,0:POKE
       1010,191:POKE
       1011,157:POKE
       1012,56
```

The program begins with a REMark, in line 10. This is a technique that you should use every time you write a program because it enables you to REMember when you wrote the program, and what name you gave to it. As you modify, and change it, save by adding a suffix, which will distinguish each version. As an example, the first copy of this program was saved using the name of "PASSWORD.1"; the second version was saved with the name of "PASSWORD.2", and the final copy had the name of "PASSWORD.F", enabling me to identify each stage of the program's development.

After I am satisfied with a program, I frequently will delete the earlier versions, and rename the final copy to do away with the ".F" suffix. Some times I keep each version on a working disk, so that I am able to refer to them later to refresh my memory of my thinking processes in the creation of the final product. You, too, might find this to be useful in your writing; don't forget to change the name and date in the REMark statement of the first line in your programs as you modify and change them!

Line 20 is another REM statement indicating what the normal values reside in the locations that will be POKEd in the program.

To see this with DOS active, type:

"FOR I = 1 TO 3:PRINT PEEK (1009 + I):NEXT I"

And, of course a RETURN.

This is what will appear on the screen:

191
157
56

Do a little arithmetic and discover that 1010 decimal is $3F2 hexadecimal. Now enter the monitor by typing:

"CALL -151" AND A RETURN
"3F2L" AND A RETURN.

This is what you should see:

```
3F2    - BF        ???
3F3    - 9D 38 4C  STA $4C38
```

And 18 additional lines of assembly language. We are interested only in the values of $3F2, $3F3 and $3F4, which are respectively:

$3F2 contains $BF
$3F3 contains $9D
$3F4 contains $38

PASSWORD PROTECTION

Do a little more mathematics, and convert the Hexadecimal values to decimal values, and you get:

$BF = 191
$9D = 157
$38 = 56

Which are exactly the values we found when we asked for the contents of these addresses using the command of "PRINT PEEK (1009+I)."

If you perform the same steps with DOS inactive (no disk booted), the values are somewhat different. Here is what they are in this case:

ADDRESS		VALUE	
DECIMAL	HEX	DECIMAL	HEX
1010	$3F2	3	$03
1011	$3F3	224	$E0
1012	$3F4	69	$45

Line 30 sets the screen to full width of 40 columns. For those of you who have not discovered the trick of "POKE 33,33" let us divert for a minute or so. This command sets the monitor screen to 33 columns and enables the programmer to easily edit lines, using Esc I,J,K, or M (in Applesoft) to move the cursor anywhere on the screen, and then to use the right and left arrows to position the cursor where desired in any line.

By using this command, there will not be blank spaces where the cursor passes off the right side of a line, and continues to the beginning of the next one. This well known technique (at least to experienced programmers) makes editing a snap. By adding line 30 to a program, I do not have to remember to reset the screen to 40 columns when I test run the program after editing using the POKE 33,33 command. Try this little trick, and you too will use it more often; it really does help me a great deal in editing, and it will do the same for you.

Lines 40,50,and 60 POKE other values in the decimal locations indicated. POKEing 214 with 128 locks out the [Ctrl-C] key, so the program can not be stopped by its use, and the other POKEs alter the RESET VECTOR so that whenever the RESET key is hit, the program will run!! These POKEs also change the DOS so that any command will cause the program to run. Thus trying to list the program to find the PASSWORD, by issuing the command "LIST" will be ineffective!!

Line 70 clears the screen; line 80 sets the cursor to vertical position to 12 and line 90 asks the user to input the PASSWORD chosen. Line 100 is a REMinder that the contents of the String PW$ must be the one selected!! It is here that you can pick the word you wish to be your secret password. My wife's name is Gloria, so in this case I have chosen this name to be the password for this program. Pick any name or word you feel is best for you; perhaps you might select one that is appropriate for the data in the program.

For example you might opt for "BUSINESS" for a business program or "RESEARCH" for data from some sort of study. You must make every effort to remember which word you chose, because without the correct one, you will not be able to access your own data easily. The only limitation is that the string selected be less than 250 characters long.

Any combination of alphabetic or numerical characters may be used, including spaces, but the longer the string, and the more complicated you make it, the harder it will be for you to remember and to type it correctly. So, be careful in your choice.

Line 110 looks at the word that was entered into the string PW. If the word is the correct password, the computer jumps to line 140 where the program can continue.

If, however, the word in the string is not the one selected to be the password, the program continues to line 120. Lets look at this line.

First the screen is cleared with the "HOME" command, (in Integer Basic, use "CALL-936") and the screen is set to the FLASHING mode. Next a VTAB to vertical position 12 and a HTAB 6 is issued. This will center the text to be printed.

Line 120 then prints the statement, "Sorry you don't know the password" which will flash on and off. If you would like to have the Apple's bell sound here also, just add:

```
:"FOR T=1 TO 5:PRINT CHR$(7):
 NEXT:"
```

Note that many commands can be placed into one line, as long as a colon separates each command! This reduces the overhead associated with a numbered command line in BASIC. In a short program such as this, overhead is not too important, but in long programs where memory may be limited, this factor must always be considered. Do you know how much overhead is associated with an Applesoft command line? You can find this information on page 118 of the Basic Programming Reference Manual.

The next command in line 120 resets the output to normal, and then issues a simple print command. The reason that this print command is needed is that the very next command is a DOS one, "PRINT CHR$(4);"PR#6".

DOS requires that this command be the very first in a line, but we have not placed it in the first position. By issuing this print command, we have fooled our Apple into thinking it was at the start of the line, because it was preceded by a carriage return.

Your disk will now be booted, and the program will end. The data that can follow in your program will not be available to anyone but you.

Line 140 clears the screen again, and the next lines set the vertical and horizontal positions to print "THAT IS THE CORRECT PASSWORD" and "LET'S GET TO WORK".

Line 190 resets the altered memory locations to normal. Add the rest of your program beginning with line 200.

Perhaps you would like to add another layer of protection to your program. You can do so by inserting these additional lines

```
200 HOME:VTAB 12:HTAB 12:
    INPUT"SECRET NUMBER IS
    : ";SN
210 IF SN< >123 THEN GOTO 30
```

Here again, you must remember what number you have entered into line 210. If you do not you will be unable to continue.

This program can be saved as an Exec Text file on your utilities disk, using any of the capture programs available. If you use a renumber program to begin the lines with 30000, you will be able to add the listing to any program you desire by adding this line to the original listing.

—●→

125 RETURN

Insert as the second line in your program (the first line should always be an identification line REM):

"GOSUB 30000"

And your program will GOTO 30000, ask for a password, and then return to your original listing.

Here is another trick that you might find useful. Use the password program as your Hello program. Now whenever your disk is Booted, it will be the first program run. Then add these lines:

```
195 PRINT FOR T=1 TO 50:
    NEXT T
196 PRINT
197 PRINT CHR$(4);
    "RUN XXXXXX"
```

Where "XXXXXX" represents any program you wish to run, so long as it is on the same disk. You will now have a Turnkey system that will run any program you name once the correct password (and secret number) have been used. Line 195 will cause a pause for you to read the screen. Pick different values for T in this line and select one that appeals to you. The higher the value you give T, the longer will be the pause.

In summary, a Password andTurnkey system will not provide perfect protection for your private programs, but will go a long way toward this goal. Use it as necessary with the idea of slowing down, but not totally preventing unauthorized entry into your records. That is until some one invents a padlock to fit into the center hole of our disks!!

Invest 10% & Receive 35% to 150%

That's the kind of speed return the Pascal Speedup System provides. Boosts in graphics and "heavyweight" business software for Apple II/II+ outperforms the competition!

STELLATION TWO

The Lobero Building P.O. Box 2342
Santa Barbara, Ca. 93120
(805) 966-1140 Telex 658439

BYTES & PIECES

Extra Special!
D.B. Master
$151.95

All Programs On Disk

	List	Ours
Arcade Machine	59.95 -	44.36
Bag of Tricks	39.95 -	29.56
Delta Drawing	59.95 -	44.36
General Manager 2.0	229.95 -	170.16
Graphics Magician	59.95 -	44.36
Home Accountant	74.95 -	55.46
Lisa Ed. System	119.95 -	88.76
Logo [Terrapin]	149.95 -	119.96
Mathemagic	89.95 -	66.56
Multiplan	275.00 -	203.49
Nibbles Away II	69.95 -	51.76
Screenwriter Pro.	199.95 -	147.96
Snooper Troops 1 or 2	44.95 -	33.26
Star Maze	34.95 -	25.86
System Saver fan	89.95 -	76.46
Tasc Compiler	175.00 -	129.49
Visicalc	250.00 -	199.96
Wizardry	49.95 -	36.96
SPECIALS		
Crossword Magic	49.95 -	34.96
DB Stat Pak	99.00 -	71.26
Graforth	75.00 -	54.49
Replay Card	129.95 -	108.96
Sensible Speller	125.00 -	90.49
Time Zone	99.95 -	71.96

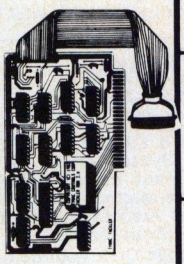
C.O.D. • Money Orders • Certified Checks • Personal Check Allow 2 Weeks
N.Y.S. Res. Add 7.25% Sales Tax • Orders Under $100.00 Add P&H
$2.00 U.S., $2.50 Canada, U.S. Funds
Foreign, Charges Only Min. P&H $6.00

• SEND FOR FREE PRICE LIST #908 •
Source TCP637

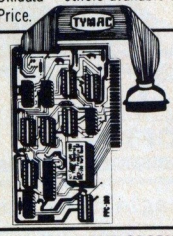
BYTES & PIECES (516) 751-2535
Box 525 Dept. C • E. Setauket, N.Y. 11733

APPLE **HARDWARE**

THE TACKLER ™ **– DUAL • MODE PARALLEL INTERFACE FOR THE APPLE® 2 BOARDS IN ONE FOR NO MORE COMPATIBILITY PROBLEMS!**



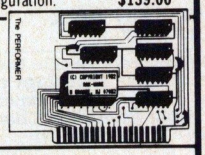
An intelligent board to provide easy control of your printer's full potential. Plus a standard parallel board at the flip of a switch – your assurance of compatibility with essentially all software for the APPLE®. Hires printing with simple keyboard commands that replace hard to use software routines. No disks to load. Special features include inverse, doubled, and rotated graphics and many text control features, available through easy keyboard or software commands. Uses Industry standard graphics commands. This is the first truly universal intelligent parallel interface! Change printers – no need to buy another board. Just plug in one of our ROM'S and you're all set. ROM'S available for Epson, C. Itoh, NEC, and Okidata – others available soon. Specify printer when ordering. Call for Price.



THE UPGRADEABLE PPC-100 PARALLEL PRINTER CARD

A Universal Centronics type parallel printer board complete with cable and connector. This unique board allows you to turn on and off the high bit so that you can access additional features in many printers. Easily upgradeable to a fully intelligent printer board with graphics and text dumps. Use with EPSON, C. ITOH, ANADEX, STAR-WRITER, NEC, OKI and others with standard Centronics configuration. **$139.00**

IF YOU WANT GRAPHICS AND FORMATTING THEN CHOOSE THE PERFORMER



for Epson, OKI, NEC 8023, C. ITOH 8510 provides resident HIRES screen dump and print formatting in firmware. Plugs into Apple slot and easy access to all printer fonts through menu with PR# command. Use with standard printer cards to add intelligence. **$49.00** specify printer.

THE MIRROR FIRMWARE FOR NOVATION APPLE CAT II®

The Data Communication Handler ROM Emulates syntax of an other popular Apple Modem product with improvements. Plugs directly on Apple CAT II Board. Supports Videx and Smarterm 80 column cards, touch tone and rotary dial, remote terminal, voice toggle, easy printer access and much more.
List $39.00 Introductory Price **$29.00**

MINI ROM BOARDS

Place your 2K program on our Mini Rom Board. Room for one 2716 EPROM. Use in any slot but zero. Only **$34.95**

DOUBLE DOS Plus

A piggy-back board that plugs into the disk-controller card so that you can switch select between DOS 3.2 and DOS 3.3 DOUBLE DOS Plus requires APPLE DOS ROMS. **$39.00**

SOFTWARE APPLE

Super Pix

Hires screendump software for the Epson, OKI, C. Itoh and Nec 8023. Use with Tymac PPC-100. Special **$19.95** (Specify Printer)

Mr. Lister **– Customer Contact Profiler & Mailer**

A Super Mail List Plus more — up to 1000 Entries on single 3.3 Disk (only 1 Drive required) — 2 second access time to any name — full sort capabilities — Dual Index Modes — supports new 9 digit Zip. Easy to follow manual — Not Copy Protected — 4 user defined tables with 26 sort selections per table — Beta tested for 6 months — user defined label generation.
Introductory Price $135. **$99.00** Dealer &: Dist. Inquiries Invited.

APPLE LINK

A communications system for the Apple® (Requires Hayes Micro Modem). Transmit and receive any type of file between APPLES®, Automatic multi-file transfer, real time clock indicating file transfer time. Complete error check. Plus conversation mode. Only one package needed for full transfers. Compatable with all DOS file types. (requires Hayes Micro Modem) **$59.00**

THE APPLE CARD/ATARI CARD

Two sided 100% plastic reference card Loaded with information of interest to all Apple and Atari owners. **$3.98**

NIBBLES AWAY II

AGAIN! Ahead of all others.

- **AUTO-LOAD PARAMETERS . . .** Free's the user from having to Manually Key in Param values used with the more popular software packages available for the Apple II.
- **EXPANDED USER MANUAL . . .** incorporates new Tutorials for all levels of expertice; Beginners Flowchart for 'where do I begin' to 'Advanced Disk Analysis' is included.
- **TRACK/SECTOR EDITOR . . .** An all new Track/Sector Editor, including the following features: Read, Write, Insert, Delete Search, and impressive Print capabilities!
- **DISK DIAGNOSTICS . .** Checks such things as: Drive Speed, Diskette Media Reliability, and Erasing Diskettes.
- **HIGHEST RATED . . .** Best back up Program in Softalk Poll (Rated 8.25 out of 10).
- **CONTINUAL UPDATES . . .** Available from Computer Applications and new listings on the source. **$69.95**

Dealer and Distributor Inquiries Invited.

VISA MasterCard

MICRO-WARE DIST. INC.
P.O. BOX 113 POMPTON PLAINS, N.J. 07444
201-838-9027

NEW Unlocked Apple® Utility Disks

Don't Blow Your Bucks on Locked-Up Uncopyable Apple Software.

NEW! Frame-Up

HI-SPEED GRAPHICS DISPLAY
BY TOM WEISHAAR

CREATE PROFESSIONAL PRESENTATIONS of intermixed hi-res, lo-res and text frames. Easy-to-use and FAST— hi-res images load in 2½-seconds! Paddles or Keyboard-advance frames in forward or reverse.

UNATTENDED SHOWS are possible with each frame individually pre-programmed to appear on the screen from 1 to 99 seconds.

TEXT SCREEN EDITOR lets you create your own b/w text "slides". Add type "live" from the keyboard during presentations if you want.

DISPLAY MODULE: Send entire presentations-on-disk to your friends and associates.

FRAME-UP: $29.50
(Includes Peeks/Pokes Chart)

Apple Mechanic

SHAPE-WRITER/BYTE-ZAP DISK
BY BERT KERSEY

SHAPE EDITOR: Keyboard-draw shapes for hi-res animation in your programs. Design proportionally-spaced typefaces with special characters. 6 fonts on the disk. Listable demos show how to use shape tables to animate games, graphics and professional Charts & Graphs.

BYTE-ZAP: Rewrite any byte on a disk for repair or alteration. Load entire sectors on the screen for inspection. Hex/Dec/Ascii displays and input. Complete instructions for making trick file names, restoring deleted files, etc.

MORE: Useful music, text and hi-res tricks for your programs. Educational documentation.

APPLE MECHANIC: $29.50
(Includes Peeks/Pokes Chart & Tip Book#5)

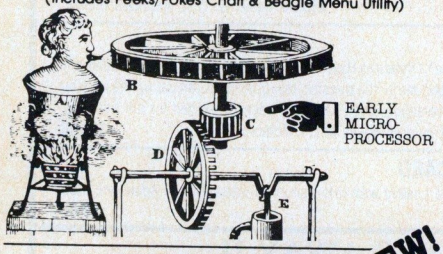
NEW! Typefaces

FOR APPLE MECHANIC

26 NEW FONTS for Apple Mechanic's Xtyper and Hi-Writer programs. Most are full 96-character fonts, large & small, of fully-editable characters. (Apple Mechanic required)

BEAGLE MENU: Use with your disks. Display only the filenames you want (e.g. only Applesoft files or only Locked files) for one-key cursor selection/execution. Space-on-disk, catalog scan, optional sector-number elimination.

TYPEFACES for Apple Mechanic: $20.00
(Includes Peeks/Pokes Chart & Beagle Menu Utility)



Beagle Bag NEW!

12-GAMES-PLUS ON ONE DISK
BY BERT KERSEY

TWELVE GREAT GAMES from the classic Beagle Bros collection— TextTrain, Slippery Digits, Wowzo, Magic Pack, Buzzword... Almost all of our "Game Pack" games, updated and re-released on one jam-packed, entertaining, unprotected disk.

COMPARE BEAGLE BAG with any one-game locked-up game disk on the market today. All 12 games are a blast, the price is right, the instructions are crystal clear, AND the disk is copyable. You can even change the programs or list them to LEARN, and see what makes them tick.

BEAGLE MENU TOO: See "Typefaces" above.

BEAGLE BAG: $29.50
(Includes Peeks/Pokes Chart & Beagle Menu Utility)

NEW! Flex Text

70-COLUMN TEXT UTILITY
BY MARK SIMONSEN

PRINT VARIABLE-WIDTH TEXT on the hi-res screens with normal Applesoft commands (including Htab 1-70). Normal, expanded & compressed text on same screen— no hardware!

ADD GRAPHICS TO TEXT or vice-versa. Run existing programs under Flex Text control. Easy to use and compatible with PLE© and GPLE.©

DOS TOOL KIT© FONT compatibility, or use Flex Text fonts. Select up to 9 fonts with ctrl-key commands. Print/List/Catalog in any style! Custom TEXT CHARACTER EDITOR included.

FLEX TEXT: $29.50
(Includes Peeks/Pokes Chart; requires monitor)

Utility City

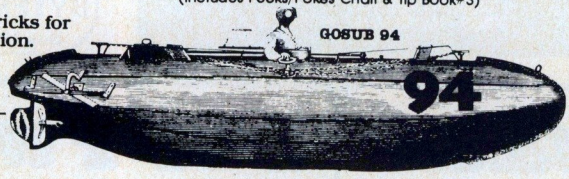
21 UTILITIES ON ONE DISK
BY BERT KERSEY

LIST FORMATTER prints each program statement on a new line. Loops indented with printer page breaks. A great de-bugger! Also...

MULTI-COLUMN catalogs for printouts, auto-post Run-number & Date in programs, put invisible commands in programs, create INVISIBLE file names, alphabetize/store info on disk, convert decimal to hex or INT to FP, renumber to 65535, append programs, dump text-screen to printer...

MORE TOO: 21 Programs Total, a best-seller!

UTILITY CITY: $29.50
(Includes Peeks/Pokes Chart & Tip Book#3)




```
10  FOR A = 1 TO 22: PRINT CHR$(ASC (MID$(
    "IJ—!IPX(T!ZPVS!TJTUFS@", A, 1))—A/A);
20  FOR B = 1 TO 4: C = PEEK(49200): NEXT B, A
```

DOS Boss

DISK COMMAND EDITOR
BY BERT KERSEY & JACK CASSIDY

RENAME COMMANDS & ERROR MESSAGES: "Catalog" can be "C"; "Syntax Error" can be "Oops" or anything you want. Protect your programs; unauthorized save-attempt can produce "Not Copyable" message. Also LIST-prevention and one-key program-run from catalog.

CUSTOMIZE DOS: Change Disk Volume heading to your message. Omit/alter catalog file codes. Fascinating documentation and tips; hours of juicy reading and Apple experiments.

ANYONE USING YOUR DISKS (booted or not) will be formatting DOS the way you designed it.

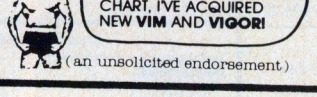
DOS BOSS: $24.00
(Includes Peeks/Pokes Chart & Tip Book#2)

Tip Disk#1

100 TIP BOOK TIPS ON DISK
BY BERT KERSEY

100 LISTABLE PROGRAMS from Beagle Bros Tip Books 1-4. Make your Apple do things its never done! All programs changeable for experimentation. Includes our Apple Command Chart: ALL Applesoft, Integer & DOS Commands!

TIP DISK#1: $20.00
(Includes Peeks/Pokes and Apple Command Charts)



(an unsolicited endorsement)

"APPLE" is a registered trade mark of You-Know-Who.

NEW! ProntoDOS

HIGH-SPEED DISK UTILITY
BY TOM WEISHAAR

HIGH-SPEED DOS! Take a look—

Function	Normal	Pronto
BLOAD HI-RES IMAGE	10 sec.	3 sec.
BSAVE HI-RES IMAGE	12 sec.	6 sec.
LOAD 60-SECTOR PROGRAM	16 sec.	4 sec.
SAVE 60-SECTOR PROGRAM	24 sec.	9 sec.
BLOAD LANGUAGE CARD	13 sec.	4 sec.
TEXT FILES	(no change)	

BOOT PRONTO-DOS or any updated normal-3.3 disk. Create new ProntoDos disks with the normal INIT command. ProntoDos is compatible with ALL DOS COMMANDS and performs normally with almost ALL programs, including CopyA.

MORE DISK SPACE: ProntoDos frees-up 15-extra-sectors per disk, almost one full track!

PRONTO-DOS: $29.50
(Includes Peeks/Pokes Chart)

Alpha Plot

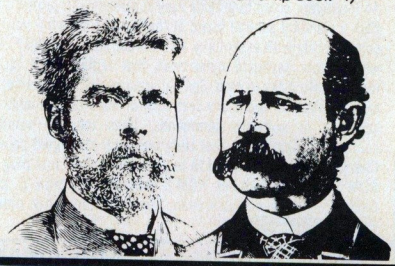
HI-RES GRAPHICS/TEXT UTILITY
BY BERT KERSEY & JACK CASSIDY

DRAW IN HI-RES, on 2 pages, using keyboard or paddles/joystick. See lines before plotting. Mixed-colors and reverse (background opposite). Fast circles, boxes and ellipses; filled or outlined.

COMPRESS HI-RES PIX to 1/3 Disk-Space. Superimpose pages or re-locate any rectangular image area anywhere on either hi-res page.

HI-RES TEXT: Proportional spacing, adjustable character size and color, upper/lower case, no tab limits, sideways typing for graphs.

ALPHA PLOT: $39.50
(Includes Peeks/Pokes Chart & Tip Book#4)



Beagle Bros™ MICRO SOFTWARE

Where to Buy Beagle Bros Disks:

MOST APPLE DEALERS carry Beagle Bros software. If yours doesn't, get on his case. Or order directly from us for IMMEDIATE SHIPMENT—

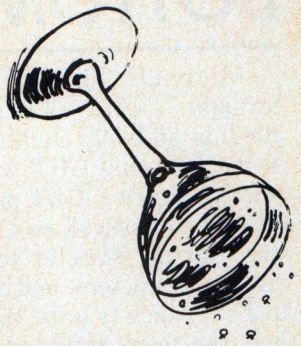
Visa/MasterCard/COD, call TOLL FREE:
Nationwide: 1-800-854-2003 ext. 827
California: 1-800-522-1500 ext. 827
Alaska/Hawaii: 1-800-854-2622 ext. 827

OR mail U.S.check, money-order or Visa/MC #'s to BEAGLE BROS, Dept. Z
4315 SIERRA VISTA / SAN DIEGO, CA 92103

Please add $1.50 First Class shipping, any size order. Overseas add $4.00. COD add $3.00. California add 6%
ALL ORDERS SHIPPED IMMEDIATELY.

A Call -A.P.P.L.E. TECHNIQUE:

Bottoms Up!



Tom Little

AS everyone knows, there are more ways to program than there are programmers. Some of them are even good or useful ways.

Contemporary with the advent of Algol in the sixties, followed by Pascal in the seventies, much attention was focused on "correct" programming practices, and how to introduce some likeness of structure and design to the process of program creation.

One of the important methodologies to come out of all this cerebral activity was the notion of "top-down" programming. Simply put, this methodology regards a computer program as a solution to a problem. The program is written by dividing the problem into smaller and smaller pieces, until the coding for each piece is trivially easy. The program structure produced by this method of attack is hierarchical. That is to say, the main program will call a set of subprograms, each of which in turn call sub-subprograms, and so on.

Top-down programming (or "stepwise refinement") has proven to be an extremely useful method for solving a wide assortment of problems.

But it is not the only way to systematically design and implement a computer program.

Enter bottom-up programming. (The phrase "Bottoms up!," by the way, was presented to me by a learned computer science professor during a lecture, and has nothing to do with programming in bars at 2 a.m. Honest.) When you program from the bottom up, you get into the nitty-gritty first and write all the dirty stuff you think you'll need to handle the task, then write the main program to allow access to these "primitive" functions.

An example may help make this approach seem less heretical. Consider your Apple. Do you suppose for a minute that Applesoft was written by stepwise refinement to form the insides of somebody's checkbook program? Or that the Apple System Monitor, all those machine language routines that make the computer go, was written just because Applesoft needed those functions? Deeper still, was the 6502 microprocessor circuit constructed to make the Monitor run?

The answer is, of course, no. The "front end" image of the Apple computer, like that of virtually every other computer in existence, is built up out of *layers.*

The 6502 chip provides a set of primitive operations which are used by the Monitor. The Monitor, in turn, provides operations which are regarded as primitives by Applesoft. And Applesoft exists only to provide you, the programmer, with just another collection of primitive (albeit powerful) operations. At every level, what is below is the "machine." Applesoft BASIC is a machine. It is, however, a machine made out of software, a Pseudo-Machine.

And Applesoft is very useful. You can solve many, many problems rather easily using the set of basic functions it gives you. So when writing an Applesoft program in top-down fashion, remember that it's not pure top-down programming, because the designers of Applesoft and the Apple computer have met you half way, coming from the bottom up.

This article is about structuring a program from the bottom up, taking the Applesoft Pseudo-Machine and building another layer on top of it.

The goal was to create a set of primitive functions that would be useful for plotting data on the hi-res screen, and have them accessible in much the way that Applesoft is accessible, all at once, with out going through menus or a hierarchical structure.

I'll go through the development process step by step, so that its advantages will be easily seen.

Step one: Fix up Applesoft

Ordinarily, this step is not part of the program development process. But there is one thing about Applesoft that has particularly bothered me. That is the INPUT statement. Is it really too much to ask to be able to read an input line into a string variable, pure and simple? If you like to plan your screen displays with care, INPUT will destroy them with its "?EXTRA IGNORED", "?REENTER," and flying question marks.

I like to write my own line input routine, that is well-behaved and attractive. It goes like this:

Figure 1

```
50100  REM
50101  REM  INPUT ROUTINE
50102  REM
50110  CALL CL:IN$ =
       ""
50120  PRINT U$;BS$;
50130  GET L1$
50140  IF L1$ = CR$
       THEN  CALL CL:
       RETURN
50150  IF L1$ = BS$
       AND  LEN (IN$)
       > 1 THEN  PRINT
       " ";BS$;BS$;:IN$
       =  LEFT$ (IN$,
       LEN (IN$) - 1):
       GOTO 50120
50151  IF L1$ = BS$
       AND  LEN (IN$)
       = 1 THEN  PRINT
       " ";BS$;BS$;:IN$
       = "": GOTO 50120
50152  IF L1$ = BS$
       THEN 50120
50160  IF L1$ < " "
       THEN ER$ = "CONTROL
       CHARACTER IN INPUT
       LINE": GOSUB 50000:
       RETURN
50170  IF L1$ >  = "
       " THEN  PRINT
       L1$;:IN$ = IN$
       + L1$
50180  GOTO 50120
```

Michael Thyng ("GETting Straight," *Call -A.P.P.L.E.* June 1982) is probably shaking his finger at me for using GET with less than pristine motives. GET makes me feel like I have control, so I use it. What this routine does is to GET successive characters, and append them to IN$, which is to be the total input line, the result returned by this subroutine. CALL CL clears the current line on the screen. CL needs to be defined as -868 (see the Applesoft manual), but I can't remember those numbers, so I write the program using variables, then look them all up at once, and assign them at the beginning of the program. U$ is the underscore character, which I use for a cursor, and BS$ is the backspace character. Line 50140 returns if a carriage return is typed. Lines 50150 to 50152 handle a backspace when typed.

In this routine, backspace is destructive, it erases the character just entered. This is different than Applesoft where you can gaily move the cursor all over the screen, and Lord only knows what's getting put in the input buffer. This way what you see is what you get, at the loss of some flexibility. Line 50160 detects a control character when typed, and goes to an error routine, which I'm assuming I'll write at line 50000. The routine will print out ER$ and do something appropriate. The redundant test in line 50170 is there so that the RETURN could be removed from line 50160. If this were done, control characters would not cause the routine to abort, but would still be filtered out of the input string, IN$.

This routine could be improved in many ways, I'm sure. But as it stands it's rather useful and easily changeable. Something like this makes a good addition to Applesoft (at least for me). It's discussed here because it relates to the idea of bottom-up programming. The input routine is a primitive operation that can be worked into any program quite easily.

A quick look at the error routine, then we'll move on to the plot program itself.

This prints the error message (previously loaded into ER$), along with the word "ERROR! ", and a beep (B$ is the bell, CHR$(7)). It then

Figure 2

```
50000  REM
50001  REM  ERROR ROUTINE
50002  REM
50010  VTAB 23: HTAB
       1: CALL CL
50020  PRINT B$;"ERROR!
       ";ER$;
50030  C = PEEK (KBD):
       IF C < 128 THEN
       50030
50031  POKE KST,0: HTAB
       1: CALL CL
50040  RETURN
```

waits (line 50030) for a keypress before continuing. This ensures that the user has read the error message and is still conscious. Line 50010 directs the printing to occur at line 23, which is a good place if you have a formatted hi-res screen display. This could be changed to have the message appear anywhere.

With the exception of line 50010, this routine could fit into any program. Bottom-up design again: Since I know I'll be needing to print out error messages, I write a primitive for it.

Step two: Listing the Primitives

Usually, this would be the first thing you'd do writing a program from the bottom up.

Really, it helps to worry about the program for a few days before writing anything down. Think about what needs to be done, and come up with ideas on how to do it. Even dream up bits and pieces of code. I know that sounds like a chaotic approach, but it helps later on to already have some details waiting to be called up.

I'd already worried some before I sat down with pencil and envelope to jot down the set of primitive operations I wanted for plotting data. The list looked about like the one below (the comments in the rightmost column I've added to explain myself a bit):

The first column is the command I want to be able to execute, and in the second column are some parameters it should be able to take. It looks rather ambitious, all set out in black and white like that. But this is the joy of programming from the bottom up: each command can be worked on separately, and tested. As individual subroutines, all those commands don't look so overwhelming.

DESTROY	[set]	erase all data in the set
APPEND	[set]	add data points to the set
DELETE	[set] [point]	remove on point from the set
DISPLAY	[set]	list all the data in the set
EDIT	[set] [point]	display a point to be changed
PLOT	[set]	plot all the points in the set
LINEPLOT	[set] [x/y]	make a linegraph of the set
BARGRAPH	[set] [x/y]	make a bargraph of the set
CLEAR		clear the hi-res screen
GRAPHICS		display graphics screen
TEXT		display text screen
ECHO	[on/off]	plot points on the screen as they are being APPENDed.
RESTART		start program over
STOP		stop program
SYMBOL	[symbol]	select the plotting symbol (circle, square, etc.)
PRINT	[set]	like display, but list to printer
SAVE	[file]	save all the data to a disk file
LOAD	[file]	load data from a disk file
SORT	[set] [x/y]	sort the data in numerical order
MEAN	[set]	display the mean value of data
SIGMA	[set]	standard deviation of data
DOS	[DOS command]	enter a DOS command
SCALE	[x] [y]	change the scaling factor
DUMP		dump screen image to printer
ORIGIN	[x] [y]	set the origin for the plot
AUTOSCALE	[set]	scale automatically, according to data values

The interesting thing to notice is that just by writing down the operations to be performed, the *data structures* used by the program have been implicitly defined. Already, by considering the operations to be performed, it is obvious that the data is to be organized into several sets, each containing a group of (x,y) points.

Data structures are very important in bottom-up design. If the program is a Pseudo-Machine, then the data structures are its architecture. Since each of the primitive operations can be written in isolation, their only communication with each other is by changing the global data structures.

Step three: Data Structures

The obvious next step, then, is to define the data structures *exactly,* since we cannot proceed to write the code for the basic commands (operations) until this is done.

The following data structures and global variables are demanded by the list of operations above:

D(NSETS,PMAX,1)
: the DATA. It is a three-dimensional array: NSETS is the number of sets we wish to have (a defined constant), and PMAX is the number of points in each set. The last index specifies the x-coordinate (0), or y (1).

NP(NSETS)
: a running count of how many points are actually being used in each set

ECHO
: 1 if echo mode on, 0 otherwise

F$	filename for data file
XS, YS	scale factors for plotting
X0, Y0	origin for plotting
SET	active set
PT	active point
SYM	the current plotting symbol

Some further refinement is necessary. For example, to simplify things, I decided to let the zeroeth point of each set go unused. If memory were a problem, this could be altered. In contrast, the sets themselves are numbered 0 through NSETS, so there are actually NSETS+1 of them.

And what is meant by scale and origin? I defined scale as the factor which the datum is multiplied by before plotting. Thus an x-scale of 2 would cause the point 10,30 to be plotted at 20,30 on the screen. Origin is the origin before scaling. An origin of 10,20 would cause a point with coordinates 10,20 to be plotted at the corner of the screen, regardless of the scale.

For plot symbols, I decided on the following:

SYM	= 0	: point (dot)
	1	: circle
	2	: square
	3	: delta (triangle)
	4	: solid circle
	5	: solid square
	6	: solid delta

This is more than just extraneous detail. The smooth operation of the final program depends entirely on how completely defined the global data structures are. You must know exactly what everything means, before you can proceed to write the code to do those primitive operations.

At this stage I realized I could write a subroutine to plot a single point. Although this isn't one of the commands listed above, it is obvious that it will need to be done many times during the program. The plot routine is lines 50400 through 50460 in the program (listing 1), and calls the subroutines from 51000 to 51640.

If you study this short subroutine, you'll see how dependent it is on the rest of the program. If SET, PT, and SYM are not set correctly (not to mention the origin and scale variables), it cannot work. The other parts of the program must provide it with meaningful data.

The advantage was that the plot routine could be thoroughly tested before writing any more of the program. All that was necessary was a main program that set the data (I even used an INPUT statement, because it was just a test program and wouldn't be needed later), then did GOSUB 50400. The test program looked something like this:

Figure 3

```
100   FOR I = 0 TO NSETS
110   FOR J = 1 TO PMAX
120 D(I,J,0) =  RND
      (1) * 280:D(I,J,1)
      =  RND (1) * 192
130   NEXT J: NEXT I
140 XS = 1:YS = 1:X0
      = 0:Y0 = 0
150   HOME : HGR : VTAB
      21
1000  INPUT SET,PT,SYM
1010  PRINT D(SET,PT,0);",
      ";D(SET,PT,1)
1020  GOSUB 50400
1030  GOTO 1000
```

The loop in lines 100-130 initializes the data with random values (NSETS and PMAX must be intialized previously, as well as KBD, KST, and CL for the error routine). The loop from 1000 to 1030 does the plotting and displays the coordinates as it does so.

This test program illustrates how a primitive can be tested more-or-less independently of the rest of the program.

Step four: Building the Machine

A Pseudo-Machine, like a hardware machine, lives by the simple sequence "fetch, decode, execute." It must get its instruction from somewhere, interpret it, and act on it. After executing one instruction in this manner, it moves on to the next. This is how the 6502 executes machine code in the Apple's memory, this is how Applesoft executes BASIC statements (either from a program or the keyboard), and this is how the plot program will run.

How to fetch a command? Well, we could have the commands in memory, like an Applesoft program. The plot program could examine memory to fetch its instruction. If we did this, we could give each command a numeric *opcode,* and it would be very easy to tell which command was intended. The problem is that this would require an editor/compiler to put those opcodes in memory in the first place, and that would be a big job. Also, a plot program should be more interactive than that. We should be able to enter commands from the keyboard in a no-nonsense manner. So the idea of the Pseudo-Machine running a stored program is placed on the back burner, for the present.

So, how do we fetch a command out of a user input line, which could be anything?

What we need is a parser. We need to be able to get an input line (GOSUB 50100, remember?), somehow peel off the first "word," and check to see if it's a valid command.

The subroutine at 50200 is a mini-parser. It takes the first word off of IN$ and places it in WD$. To this routine, anything with a space or comma after it is a word. It's a pretty intelligent routine, and will strip off leading and trailing blanks, and leave IN$ with the first word removed so that it can return the *second* word the next time it's called (and so on).

—●→

BOTTOMS UP!

As long as we have to write a little parsing routine anyway (I call it GETWORD), it might as well help us out by giving us some idea what it did. It returns a status code in the variable RESULT, like so:

RESULT =
- 0: WD$ is numeric, more words left in IN$
- 1: WD$ is numeric, last word in IN$
- 2: WD$ is non-numeric, more words left in IN$
- 3: WD$ is non-numeric, last word in IN$

This RESULT tells us whether or not we got a number, and whether we've reached the end of the input string. This will come in very handy when we have to parse a command's parameters.

So that's fetching: GOSUB 50100: GOSUB 50200. After that sequence, the first word of the input string is in WD$, and we can check to see if it's a command.

How? Well, that's the next phase: decoding. Fortunately, it has an easy solution. We just make an array of strings, with all our commands in it. Then we use a FOR loop to compare WD$ with each element in the array. When we find a match, we save the index into the array at that point. We come out of the loop having translated WD$ into a numeric code for the command.

But what good is that? Well, onto the the next phase: execution. Once we've got a command in the form of a number, we can use Applesoft's ON x GOSUB statement to execute a unique subroutine for each command.

That's all there is to it. We just need to write a subroutine for each command we want to do, and the whole job will be done. The subroutines are responsible for parsing and decoding their own parameters. All the "main program" does is fetch, decode, execute. Lines 220–222 in listing 1 intialize the command string array, and lines 1000–1070 are the main loop of the program.

At this point, the program deserves a name. I called it DPCI, Data Plot Command Interpreter. It really is an interpreter, in the same sense that Applesoft is.

With the main loop taken care of, we can write subroutines to execute the primitives at our leisure. This is the fifth and last step.

Step five: Doing the Work

Everything is ready now to write the actual coding for the subroutines to perform the execution of the various DPCI commands.

Now it may seem like it's not honest bottom-up programming to write that main control loop *before* the subroutines, and I guess in truth it really isn't. But the control loop is trivial, and it doesn't in any way define or limit the user's access to the primitives (except for the interactive/stored program decision), like the main program in a top-down design does. So we might as well put it in early, so it can help the development and debugging along.

The first subroutine I wrote was APPEND, so that I could get data into the system and test the other subroutines.

Listing 1

```
]LIST

 10  REM
 ************************************
 * DATA PLOT COMMAND INTERPRETER *
 * ( D P C I )  - BY TOM LITTLE  *
 * CALL -A.P.P.L.E. : JAN. 1983  *
 ************************************

 100  REM
 101  REM  DEFINE CONSTANTS
 102  REM
 110 D$ =  CHR$ (13) +  CHR$ (4):B
      $ =  CHR$ (7):BS$ =  CHR$ (8
      ):CR$ =  CHR$ (13):U$ =  CHR$
      (95)
 120 CL =  - 868:GM =  - 16304:TM =
       - 16303:KBD =  - 16384:KST =
       - 16368:BM = 35
 130 NSETS = 5:PMAX = 20
 140 PR = 1
 150  ONERR  GOTO 50800
 200  REM
 201  REM  SET UP DATA STRUCTURES
 202  REM
 210  DIM D(NSETS,PMAX,1),NP(NSETS
      ),DUMMY(PMAX,1)
 220 NC = 25: DIM COMM$(NC)
 221  FOR I = 0 TO NC: READ COMM$(
      I): NEXT
 222  DATA       DESTROY,APPEND,DEL
      ETE,DISPLAY,EDIT,PLOT,LINEPL
      OT,BARGRAPH,CLEAR,GRAPHICS,T
      EXT,ECHO,RESTART,STOP,SYMBOL
      ,PRINT,SAVE,LOAD,SORT,MEAN,S
      IGMA,DOS,SCALE,DUMP,ORIGIN,A
      UTOSCALE
 230  DIM SYM$(3)
 231  FOR I = 0 TO 3: READ SYM$(I)
      : NEXT
 232  DATA POINT,CIRCLE,SQUARE,DEL
      TA
 300  REM  INITIAL VALUES
 310 P = 1:ECHO = 0:F$ = "WORKFILE
      (DPCI)":XS = 1:YS = 1:X0 = 0
      :Y0 = 0
 320 SET = 0:PT = 0
 400  REM  STARTUP
 410  TEXT : HOME : HTAB 18: INVERSE
      : PRINT "DPCI": NORMAL : PRINT
      : PRINT : HTAB 5: PRINT "DAT
      A PLOT COMMAND INTERPRETER":
       PRINT : HTAB 9: PRINT "BY T
      OM LITTLE MAY 1982"
 420  VTAB 20: PRINT "INTERACTIVE
      MODE? (Y/N) ";: GET IC$: IF
      IC$ <  > "N" THEN 450
 430  PRINT "NO": PRINT "NAME OF C
      OMMAND FILE: ";: GOSUB 50100
      :CF$ = IN$:CF = 1
 440  PRINT D$;"OPEN ";CF$: PRINT
      D$;"READ ";CF$
 450  HGR : GOSUB 50500: HCOLOR= 3

 998  REM
 999  REM  ------------------
 1000  REM
 1001  REM  MAIN CONTROL LOOP
 1002  REM
```

```
1010  VTAB 22: HTAB 1: PRINT "COM
     MAND: ";: GOSUB 50100
1020  IF IN$ = "" THEN 1010
1030  GOSUB 50200:COMM =  - 1
1040  FOR I = 0 TO NC: IF COMM$(I
     ) = WD$ THEN COMM = I
1050  NEXT
1055  IF COMM =  - 1 THEN ER$ = "
     BAD COMMAND": GOSUB 50000
1060  ON COMM + 1 GOSUB 10000,101
     00,10200,10300,10400,10500,1
     0600,10700,10800,10900,11000
     ,11100,11200,11300,11400,115
     00,11600,11700,11800,11900,1
     2000,12100,12200,12300,12400
     ,12500
1065 F =  FRE (0)
1070  GOSUB 50500: GOTO 1000
9998  REM
9999  REM  ----------------------

10000  REM
10001  REM  DPPM COMMANDS
10002  REM
10003  REM
10004  REM  DESTROY
10005  REM
10010  IF RESULT = 3 THEN 10040
10020  GOSUB 50200:LS =  ASC (WD$
     ) -  ASC ("A"): IF LS ) NSET
     S OR LS ( 0 THEN ER$ = "(DES
     TROY) BAD SET": GOSUB 50000:
      RETURN
10030 SET = LS
10040 NP(SET) = 0
10050  RETURN
10100  REM
10101  REM  APPEND
10102  REM
10110  IF RESULT = 3 THEN 10140
10120  GOSUB 50200:LS =  ASC (WD$
     ) -  ASC ("A"): IF LS ) NSET
     S OR LS ( 0 THEN ER$ = "(APP
     END) BAD SET": GOSUB 50000: RETURN
10130 SET = LS
10140  GOSUB 50500
10150  IF NP(SET) = PMAX THEN ER$
      = "(APPEND) TOO MANY POINTS
     ": GOSUB 50000: RETURN
10160  VTAB 22: HTAB 1: PRINT "AP
     PEND: ";: GOSUB 50100: IF IN
     $ = "" THEN  RETURN
10170  GOSUB 50200: IF RESULT (  )
     0 THEN ER$ = "(APPEND) BAD P
     ARAMETER": GOSUB 50000: GOTO
     10140
10175 NP(SET) = NP(SET) + 1
10180 D(SET,NP(SET),0) =  VAL (WD
     $): GOSUB 50200: IF RESULT (
      ) 1 THEN ER$ = "(APPEND) BA
     D PARAMETER": GOSUB 50000: GOTO
     10140
10190 D(SET,NP(SET),1) =  VAL (WD
     $): IF ECHO THEN PT = NP(SET
     ): GOSUB 50400
10191  GOSUB 50500: GOTO 10140
10200  REM
10201  REM  DELETE
10202  REM
10210  GOSUB 50200: IF RESULT = 1
      THEN  GOTO 10240
10220  IF RESULT (  ) 2 THEN ER$ =
     "(DELETE) BAD PARAMETER": GOSUB
     50000: RETURN
10230 LS =  ASC (WD$) -  ASC ("A"
     ): IF LS ) NSETS OR LS ( 0 THEN
     ER$ = "(DELETE) BAD SET": GOSUB
     50000: RETURN
10239  GOSUB 50200
10240  IF RESULT (  ) 1 THEN ER$ =
     "(DELETE) BAD PARAMETER": GOSUB
     50000: RETURN
10241 SET = LS
10250 PT =  VAL (WD$): IF PT ( 1 OR
     PT ) NP(SET) THEN ER$ = "(DE
     LETE) BAD POINT": GOSUB 5000
     0: RETURN
10260  FOR LP = PT TO NP(SET) - 1
     :D(SET,LP,0) = D(SET,LP + 1,
     0):D(SET,LP,1) = D(SET,LP +
     1,1): NEXT
10270 NP(SET) = NP(SET) - 1
10280  RETURN
10300  REM
10301  REM  DISPLAY
10302  REM
10310  IF RESULT = 3 THEN  TEXT :
      POKE BM,20: HOME : FOR LS =
     0 TO NSETS: GOSUB 10350: NEXT
     : POKE GM,0: POKE BM,24: RETURN

10320  IF RESULT (  ) 2 THEN ER$ =
     "(DISPLAY) BAD PARAMETER": GOSUB
     50000: RETURN
10330  GOSUB 50200:LS =  ASC (WD$
     ) -  ASC ("A"): IF LS ) NSET
     S OR LS ( 0 THEN ER$ = "(DIS
     PLAY) BAD SET": GOSUB 50000:
      RETURN
10340 SET = LS: TEXT : POKE BM,20
     : HOME : GOSUB 10350: POKE G
     M,0: POKE BM,24: RETURN
10350  PRINT : PRINT "SET "; CHR$
     (LS +  ASC ("A"));":"
10359  IF NP(LS) ( 2 THEN 10380
10360  FOR LP = 1 TO  INT (NP(LS)
     / 2) * 2 - 1 STEP 2
10370  HTAB 1: PRINT "(";LP;") ";
     D(LS,LP,0);", ";D(LS,LP,1);
10371  HTAB 20: PRINT "(";LP + 1;
     ") ";D(LS,LP + 1,0);", ";D(L
     S,LP + 1,1)
10372  NEXT
10380  IF NP(LS) )  INT (NP(LS) /
     2) * 2 THEN  PRINT "(";NP(LS
     );") ";D(LS,NP(LS),0);", ";D
     (LS,NP(LS),1)
10390  GET LC$: RETURN
10400  REM
10401  REM  EDIT
10402  REM
10410  GOSUB 50200: IF RESULT = 1
      THEN  GOTO 10440
10420  IF RESULT (  ) 2 THEN ER$ =
     "(EDIT) BAD PARAMETER": GOSUB
     50000: RETURN
10430 LS =  ASC (WD$) -  ASC ("A"
     ): IF LS ) NSETS OR LS ( 0 THEN
     ER$ = "(EDIT) BAD SET": GOSUB
     50000: RETURN
10439  GOSUB 50200
10440  IF RESULT (  ) 1 THEN ER$ =
     "(EDIT) BAD PARAMETER": GOSUB
     50000: RETURN
10450 PT =  VAL (WD$): IF PT ( 1 OR
     PT ) NP(SET) THEN ER$ = "(ED
     IT) BAD POINT": GOSUB 50000:
      RETURN
10451 SET = LS: GOSUB 50500
10460  VTAB 23: HTAB 2: PRINT "("
     ;PT;") ";D(SET,PT,0);", ";D(
     SET,PT,1)
10470  VTAB 22: PRINT "EDIT: ";: GOSUB
     50100: IF IN$ = "" THEN  RETURN

10480  GOSUB 50200: IF RESULT (  )
     0 THEN ER$ = "(EDIT) BAD PAR
     AMETER": GOSUB 50000: RETURN
```

The other subroutines were added one by one, more or less in the order in which I originally jotted them down. Each one could be tested in the *working program* as it was written. This is very satisfying, because you can see results quickly and enjoy the augmenting of the program, rather than investing a lot of time and work, only to have something inexplicable go wrong when you try to run the finished creation.

To illustrate how the subroutines were written to work into the DPCI "architecture," take a look at the code for LINEPLOT (listing 1, lines 10600–10700).

The first thing it does is parse its parameters. LINEPLOT expects a command line in one of the following forms:

LINEPLOT
LINEPLOT [set]
LINEPLOT [set] [x/y]

In the syntax language used above, [set] stands for a letter designating the set of data points to be plotted. "A" means the first set, "B" the second, etc. This is less confusing than having to give a number for each set, since the points within a set are referred to by number. The [x/y] parameter allows you to select whether the points should be connected in numerical order by the x coordinate or by the y coordinate. This parameter can be either "X" or "Y." The former will produce a broken horizontal line, the latter a broken vertical line.

A decision was made to let "X" be the default mode. Of course the current working set is the default set. Notice that in order to get a "Y" style plot, the set must be specified too.

Line 10609 checks for the first example given above (i.e. no parameters). Remember RESULT=3 indicates no more words in the string (after the command word was extracted). If this is the case, we set up the mode to be "X," skipping over the set selection part (lines 10610–10630).

Line 10610 calls GETWORD, and we check the word returned (WD$) to see if it's a valid set. If not, we flag the error and abort the LINEPLOT. In 10630 we check to see if there are more parameters. If not (RESULT =3), we assume "X" mode and continue. Otherwise, we call GETWORD again to pick up the mode (lines 10635 and 10640).

At this point, we're ready to do the lineplot. First we must sort the data in the set, so that we can connect the points in the correct order. But since the user may not want the data rearranged, we sort it in a temporary array called DUMMY. The data from the working set is transferred to DUMMY by line 10650, which also calls SORT (GOSUB 50600). Line 10670 plots all the points, using the current symbol, and lines 10680–10691 connect the dots!

The RETURN in line 10691 takes us back to the main loop, where we fetch the next command.

```
10490 D(SET,PT,0) =  VAL (WD$): GOSUB
      50200: IF RESULT <  > 1 THEN
      ER$ = "(EDIT) BAD PARAMETER"
      : GOSUB 50000: RETURN
10495 D(SET,PT,1) =  VAL (WD$): RETURN

10500  REM
10501  REM  PLOT
10502  REM
10510  IF RESULT = 3 THEN 10550
10520  GOSUB 50200: IF RESULT <  >
      3 THEN ER$ = "(PLOT) BAD PAR
      AMETER": GOSUB 50000: RETURN

10530 LS =  ASC (WD$) -  ASC ("A"
      ): IF LS > NSETS OR LS < 0 THEN
      ER$ = "(PLOT) BAD SET": GOSUB
      50000: RETURN
10540 SET = LS: GOSUB 50500
10550  IF NP(SET) > 0 THEN  FOR P
      T = 1 TO NP(SET): GOSUB 5040
      0: NEXT : RETURN
10560 ER$ = "NO POINTS TO PLOT": GOSUB
      50000: RETURN
10600  REM
10601  REM  LINE PLOT
10602  REM
10609  IF RESULT = 3 THEN WD$ = "
      X": GOTO 10640
10610  GOSUB 50200
10620 LS =  ASC (WD$) -  ASC ("A"
      ): IF LS > NSETS OR LS < 0 THEN
      ER$ = "(LINEPLOT) BAD SET": GOSUB
      50000: RETURN
10630 SET = LS: GOSUB 50500: IF R
      ESULT = 3 THEN WD$ = "X": GOTO
      10640
10635  GOSUB 50200
10640 MODE$ = WD$: IF MODE$ <  >
      "Y" THEN MODE$ = "X"
10650  FOR LP = 1 TO NP(SET):DUMM
      Y(LP,0) = D(SET,LP,0):DUMMY(
      LP,1) = D(SET,LP,1): NEXT : GOSUB
      50600
10660  IF NP(SET) < 1 THEN  RETURN

10670  FOR FT = 1 TO NP(SET): GOSUB
      50400: NEXT
10680 X = (DUMMY(1,0) - X0) * XS:
      Y = 159 - (DUMMY(1,1) - Y0) *
      YS: IF X >  = 0 AND Y >  = 0
       AND X <  = 279 AND Y <  = 1
      59 THEN  HPLOT X,Y
10690  FOR I = 1 TO NP(SET):X = (
      DUMMY(I,0) - X0) * XS:Y = 15
      9 - (DUMMY(I,1) - Y0) * YS: IF
      X >  = 0 AND Y >  = 0 AND X <
       = 279 AND Y <  = 159 THEN  HPLOT
       TO X,Y
10691  NEXT : RETURN
10700  REM
10701  REM  BAR GRAPH
10702  REM
10709  IF RESULT = 3 THEN WD$ = "
      X": GOTO 10740
10710  GOSUB 50200
10720 LS =  ASC (WD$) -  ASC ("A"
      ): IF LS > NSETS OR LS < 0 THEN
      ER$ = "(BARGRAPH) BAD SET": GOSUB
      50000: RETURN
10730 SET = LS: GOSUB 50500: IF R
      ESULT = 3 THEN WD$ = "X": GOTO
      10740
10735  GOSUB 50200
10740 MODE$ = WD$: IF MODE$ <  >
      "Y" THEN MODE$ = "X"
10750  FOR LP = 1 TO NP(SET):DUMM
      Y(LP,0) = D(SET,LP,0):DUMMY(
      LP,1) = D(SET,LP,1): NEXT : GOSUB
      50600
10760  IF NP(SET) < 2 THEN  RETURN
```

```
10770  IF MODE$ = "Y" THEN 10780
10771 D = DUMMY(1,0) - DUMMY(2,0)

10775  FOR LP = 1 TO NP(SET)
10776 X1 = (DUMMY(LP,0) - X0 - D /
     2) * XS:X2 = X1 + D * XS:Y =
     159 - (DUMMY(LP,1) - Y0) * Y
     S: IF X1 < 0 OR X1 > 279 OR
     X2 < 0 OR X2 > 279 OR Y < 0 OR
     Y > 159 THEN ER$ = "(BARGRAP
     H) POINT OFF SCREEN": GOSUB
     50000: GOTO 10778
10777  HPLOT X1,159 TO X1,Y TO X2
     ,Y TO X2,159
10778  NEXT
10779  RETURN
10780  REM  Y MODE
10781 D = DUMMY(1,1) - DUMMY(2,1)

10785  FOR LP = 1 TO NP(SET)
10786 Y1 = 159 - (DUMMY(LP,1) - Y
     0 - D / 2) * YS:Y2 = Y1 - D *
     YS:X = (DUMMY(LP,0) - X0) *
     XS: IF Y1 < 0 OR Y1 > 159 OR
     Y2 < 0 OR Y2 > 159 OR X < 0 OR
     X > 279 THEN ER$ = "(BARGRAP
     H) POINT OFF SCREEN": GOSUB
     50000: GOTO 10788
10787  HPLOT 0,Y1 TO X,Y1 TO X,Y2
     TO 0,Y2
10788  NEXT
10789  RETURN
10800  REM
10801  REM  CLEAR
10802  REM
10810  HGR : RETURN
10900  REM
10901  REM  GRAPHICS
10902  REM
10910  POKE GM,0: RETURN
11000  REM
11001  REM  TEXT
11002  REM
11010  POKE TM,0: RETURN
11100  REM
11101  REM  ECHO
11102  REM
11110 ECHO = 1: IF RESULT = 3 THEN
     RETURN
11120  GOSUB 50200: IF WD$ = "OFF
     " THEN ECHO = 0
11130  RETURN
11200  REM
11201  REM  RESTART
11202  REM
11210  VTAB 23: HTAB 1: PRINT "RE
     START PROGRAM? (Y/N) ";: GET
     LC$: IF LC$ = "Y" THEN  PRINT
     D$"CLOSE": RUN
11220  RETURN
11300  REM
11301  REM  STOP
11302  REM
11310  VTAB 23: HTAB 1: PRINT "ST
     OP PROGRAM? (Y/N) ";: GET LC
     $: IF LC$ = "Y" THEN  PRINT
     D$"CLOSE": TEXT : HOME : PRINT
     "PROGRAM STOPPED.": END
11400  REM
11401  REM  SYMBOL
11402  REM
11410 LS = 0: GOSUB 50200: IF RES
     ULT = 3 THEN 11440
11420  IF WD$ = "SOLID" THEN LS =
     3
11430  GOSUB 50200: IF WD$ = "POI
     NT" THEN LS = 0
11440 L2 =  - 1: FOR LI = 0 TO 3:
     IF SYM$(LI) = WD$ THEN L2 = LI
11450  NEXT : IF L2 <  > - 1 THEN
     SYM = LS + L2
11460  RETURN
11500  REM
11501  REM  PRINT
11502  REM
11510  IF RESULT = 3 THEN  PRINT
     D$;"PR#";PR: FOR LS = 0 TO N
     SETS: GOSUB 11550: NEXT : PRINT
     D$;"PR#0": RETURN
11520  IF RESULT <  > 2 THEN ER$ =
     "(PRINT) BAD PARAMETER": GOSUB
     50000: RETURN
11530  GOSUB 50200:LS =  ASC (WD$
     ) -  ASC ("A"): IF LS > NSET
     S OR LS < 0 THEN ER$ = "(PRI
     NT) BAD SET": GOSUB 50000: RETURN

11540 SET = LS: PRINT D$;"PR#";PR
     : GOSUB 11550: PRINT D$;"PR#
     0": RETURN
11550  PRINT : PRINT "SET "; CHR$
     (LS +  ASC ("A"));":"
11559  IF NP(LS) = 0 THEN  RETURN

11560  FOR LP = 1 TO NP(LS)
11570 OUT$ = "(" +  STR$ (LP) + "
     ) " +  STR$ (D(LS,LP,0)) + "
     , " +  STR$ (D(LS,LP,1)): FOR
     LI = 1 TO 20 -  LEN (OUT$):O
     UT$ = OUT$ + " ": NEXT LI
11580  PRINT OUT$;: IF  INT (LP /
     4) * 4 = LP THEN  PRINT
11590  NEXT LP: PRINT : RETURN
11600  REM
11601  REM  SAVE
11602  REM
11610  IF RESULT = 3 THEN 11630
11620  GOSUB 50200:F$ = WD$
11630  PRINT D$;"OPEN ";F$: PRINT
     D$;"DELETE ";F$: PRINT D$;"O
     PEN ";F$: PRINT D$;"WRITE ";
     F$
11640  FOR LI = 0 TO NSETS
11650  PRINT NP(LI)
11660  FOR LJ = 1 TO NP(LI): PRINT
     D(LI,LJ,0);",";D(LI,LJ,1): NEXT
     LJ
11670  NEXT LI
11680  PRINT D$;"CLOSE ";F$: RETURN

11700  REM
11701  REM  LOAD
11702  REM
11710  IF RESULT = 3 THEN 11730
11720  GOSUB 50200:F$ = WD$
11730  PRINT D$;"OPEN ";F$: PRINT
     D$;"READ ";F$
11740  FOR LI = 0 TO NSETS
11750  INPUT NP(LI)
11760  FOR LJ = 1 TO NP(LI): INPUT
     D(LI,LJ,0),D(LI,LJ,1): NEXT
     LJ
11770  NEXT LI
11780  PRINT D$;"CLOSE ";F$: IF C
     F THEN  PRINT D$;"READ ";CF$

11790  RETURN
11800  REM
11801  REM  SORT
11802  REM
11809  IF RESULT = 3 THEN WD$ = "
     X": GOTO 11840
11810  GOSUB 50200: IF RESULT = 3
     THEN 11840
11820 LS =  ASC (WD$) -  ASC ("A"
     ): IF LS > NSETS OR LS < 0 THEN
     ER$ = "(SORT) BAD SET": GOSUB
     50000: RETURN
```

```
11830 SET = LS: GOSUB 50500: GOSUB
      50200
11840 MODE$ = WD$: IF MODE$ <  >
      "Y" THEN MODE$ = "X"
11850  FOR LP = 1 TO NP(SET):DUMM
      Y(LP,0) = D(SET,LP,0):DUMMY(
      LP,1) = D(SET,LP,1): NEXT : GOSUB
      50600
11860  FOR LP = 1 TO NP(SET):D(SE
      T,LP,0) = DUMMY(LP,0):D(SET,
      LP,1) = DUMMY(LP,1): NEXT : RETURN
11900  REM
11901  REM   MEAN
11902  REM
11910  IF RESULT = 3 THEN 11940
11920  GOSUB 50200: IF RESULT <  >
      3 THEN ER$ = "(MEAN) BAD PAR
      AMETER": GOSUB 50000: RETURN

11930 LS =  ASC (WD$) -  ASC ("A"
      ): IF LS < 0 OR LS > NSETS THEN
      ER$ = "(MEAN) BAD SET": GOSUB
      50200: RETURN
11931 SET = LS: GOSUB 50500:
11940  GOSUB 50700
11950  VTAB 23: HTAB 1: PRINT " M
      EAN = ";MX;", ";MY;
11960  GET LC$: VTAB 23: HTAB 1: CALL
      CL: RETURN
12000  REM
12001  REM  SIGMA - STANDARD DEVI
      ATION
12002  REM
12010  IF RESULT = 3 THEN 12040
12020  GOSUB 50200: IF RESULT <  >
      3 THEN ER$ = "(SIGMA) BAD PA
      RAMETER": GOSUB 50000: RETURN

12030 LS =  ASC (WD$) -  ASC ("A"
      ): IF LS < 0 OR LS > NSETS THEN
      ER$ = "(SIGMA) BAD SET": GOSUB
      50200: RETURN
12031 SET = LS: GOSUB 50500
12040  GOSUB 50700
12050 S1 = 0:S2 = 0: FOR LP = 1 TO
      NP(SET):S1 = S1 + (MX - D(SE
      T,LP,0)) ^ 2:S2 = S2 + (MY -
      D(SET,LP,1)) ^ 2
12060 S1 =  SQR (S1 / NP(SET)):S2
      =  SQR (S2 / NP(SET))
12070  VTAB 23: HTAB 1: PRINT " S
      .D. = ";S1;", ";S2;
12080  GET LC$: VTAB 23: HTAB 1: CALL
      CL: RETURN
12100  REM
12101  REM  DOS COMMAND
12102  REM
12110  POKE BM,20: HOME
12120  PRINT D$;IN$
12130  IF CF THEN  PRINT D$;"READ";CF$
12140  POKE BM,24: RETURN
12200  REM
12201  REM  SCALE
12202  REM
12210  GOSUB 50200: IF RESULT <  >
      0 THEN ER$ = "(SCALE) BAD PA
      RAMETER": GOSUB 50000: RETURN

12220 XS =  VAL (WD$): GOSUB 5020
      0: IF RESULT <  > 1 THEN  THEN
      ER$ = "(SCALE) BAD PARAMETER
      ": GOSUB 50000: RETURN
12230 YS =  VAL (WD$): RETURN
12300  REM
12301  REM  DUMP
12302  REM
12310 ER$ = "NOT CONFIGURED FOR D
      UMP": GOSUB 50000: RETURN
12400  REM
12401  REM  ORIGIN
12402  REM
12410  GOSUB 50200: IF RESULT <  >
      0 THEN ER$ = "(ORIGIN) BAD P
      ARAMETER": GOSUB 50000: RETURN

12420 X0 =  VAL (WD$): GOSUB 5020
      0: IF RESULT <  > 1 THEN ER$
      = "(ORIGIN) BAD PARAMETER":
      GOSUB 50000: RETURN
12430 Y0 =  VAL (WD$): RETURN
12500  REM
12501  REM  AUTOSCALE
12502  REM
12510  IF RESULT = 3 THEN 12540
12520  GOSUB 50200: IF RESULT <  >
      3 THEN ER$ = "(AUTOSCALE) BA
      D PARAMETER": GOSUB 50000: RETURN

12530 LS =  ASC (WD$) -  ASC ("A"
      ): IF LS < 0 OR LS > NSETS THEN
      ER$ = "(AUTOSCALE) BAD SET":
      GOSUB 50000: RETURN
12531 X1 = D(LS,1,0):X2 = X1:Y1 =
      D(LS,1,1):Y2 = Y1
12535 SET = LS: GOSUB 12550: RETURN

12540 X1 = D(SET,1,0):X2 = X1:Y1 =
      D(SET,1,1):Y2 = Y1
12545  FOR LS = 0 TO NSETS: GOSUB
      12550: NEXT : RETURN
12550  IF NP(LS) < 1 THEN  RETURN

12551  FOR LP = 1 TO NP(LS): IF D
      (LS,LP,0) < X1 THEN X1 = D(L
      S,LP,0)
12552  IF D(LS,LP,1) < Y1 THEN Y1
      = D(LS,LP,1)
12553  IF D(LS,LP,0) > X2 THEN X2
      = D(LS,LP,0)
12554  IF D(LS,LP,1) > Y2 THEN Y2
      = D(LS,LP,1)
12555  NEXT
12560 X0 = X1:Y0 = Y1
12570  IF X2 > X1 THEN XS = 279 /
      (X2 - X1)
12580  IF Y2 > Y1 THEN YS = 159 /
      (Y2 - Y1)
12590  RETURN
49996  REM
49997  REM  ---------------------
      -
49998  REM
49999  REM   SUPPORT ROUTINES
50000  REM
50001  REM  ERROR ROUTINE
50002  REM
50010  VTAB 23: HTAB 1: CALL CL
50020  PRINT B$;"ERROR! ";ER$;
50030 C =  PEEK (KBD): IF C < 128
      THEN 50030
50031  POKE KST,0: HTAB 1: CALL CL
50040  RETURN
50100  REM
50101  REM  INPUT ROUTINE
50102  REM
50110  CALL CL:IN$ = ""
50120  PRINT U$;BS$;
50130  GET L1$
50140  IF L1$ = CR$ THEN  CALL CL
      : RETURN
```

```
50150  IF L1$ = BS$ AND  LEN (IN$
     ) > 1 THEN  PRINT " ";BS$;BS
     $;:IN$ =  LEFT$ (IN$, LEN (I
     N$) - 1): GOTO 50120
50151  IF L1$ = BS$ AND  LEN (IN$
     ) = 1 THEN  PRINT " ";BS$;BS
     $;:IN$ = "": GOTO 50120
50152  IF L1$ = BS$ THEN 50120
50160  IF L1$ < " " THEN ER$ = "C
     ONTROL CHARACTER IN INPUT LI
     NE": GOSUB 50000: RETURN
50170  IF L1$ >  = " " THEN  PRINT
     L1$;:IN$ = IN$ + L1$
50180  GOTO 50120
50200  REM
50201  REM  GETWORD
50202  REM
50210 WD$ = "":RESULT = 0
50220  IF IN$ = "" THEN ER$ = "(G
     ETWORD) EMPTY STRING": GOSUB
     50000:RESULT = 3: RETURN
50230 LC$ =  LEFT$ (IN$,1)
50231  IF  LEN (IN$) > 1 THEN IN$
     =  RIGHT$ (IN$, LEN (IN$) -
     1): GOTO 50240
50232 IN$ = "": REM  IF LEN(IN$)=
     1
50240  REM  REMOVE LEADING BLANKS
     -
50241  IF LC$ = " " THEN 50220
50250  IF LC$ = " " OR LC$ = "," THEN
     50310
50260 WD$ = WD$ + LC$
50270  IF (LC$ < "0" OR LC$ > "9"
     ) AND LC$ <  > "-" AND LC$ <
     > "." THEN RESULT = 2
50280 LC$ =  LEFT$ (IN$,1)
50281  IF  LEN (IN$) > 1 THEN IN$
     =  RIGHT$ (IN$, LEN (IN$) -
     1): GOTO 50290
50282 IN$ = "": REM   IF LEN(IN$)
     =  1
50290  IF IN$ <  > "" THEN 50250
50300 WD$ = WD$ + LC$:RESULT = RE
     SULT + 1: RETURN : REM  EN
     D OF IN$ REACHED
50310  REM  REMOVE TRAILING BLANK
     S -
50311 LC$ =  LEFT$ (IN$,1): IF LC
     $ <  > " " THEN  RETURN
50312 IN$ =  RIGHT$ (IN$, LEN (IN
     $) - 1): IF IN$ = "" THEN 50
     300
50313  GOTO 50311
50400  REM
50401  REM  PLOT
50402  REM
50410 X = (D(SET,PT,0) - X0) * XS

50420 Y = 159 - (D(SET,PT,1) - Y0
     ) * YS
50430  IF X < 0 OR Y < 0 OR X > 2
     79 OR Y > 159 THEN ER$ = "(P
     LOT) POINT OFF SCREEN": GOSUB
     50000: RETURN
50440 LS = SYM: IF X < 2 OR Y < 2
     OR X > 277 OR Y > 157 THEN
     LS = 0
50450  ON LS + 1 GOSUB 51000,5110
     0,51200,51300,51400,51500,51 600
50460  RETURN
50500  REM
50501  REM  REDRAW SCREEN
50510  VTAB 22: HTAB 1: CALL CL: VTAB
     23: CALL CL
50520  VTAB 21: HTAB 1: CALL CL: PRINT
     "FILE: ";F$;: HTAB 25: PRINT
     "  SET "; CHR$ ( ASC ("A") +
     SET);: HTAB 36: INVERSE : PRINT
     "DPCI";: NORMAL
50530  VTAB 24: HTAB 1: CALL CL: IF
     ECHO THEN  PRINT "ECHO";
50539 S0 = SYM
50540  HTAB 7: PRINT "#POINTS=";N
     P(SET);: HTAB 20: PRINT "SYM
     BOL=";: IF SYM > 3 THEN  PRINT
     "SOLID ";:S0 = SYM - 3
50550  PRINT SYM$(S0);: CALL CL
50560  RETURN
50600  REM
50601  REM  SORT
50602  REM
50603  VTAB 23: HTAB 1: PRINT "PL
     EASE BE PATIENT..."
50610  IF NP(SET) < 2 THEN  RETURN

50620  FOR SI = 1 TO NP(SET) - 1
50630  FOR SJ = SI + 1 TO NP(SET)

50640  IF MODE$ = "X" AND DUMMY(S
     J,0) < DUMMY(SI,0) THEN T0 =
     DUMMY(SI,0):T1 = DUMMY(SI,1)
     :DUMMY(SI,0) = DUMMY(SJ,0):D
     UMMY(SI,1) = DUMMY(SJ,1):DUM
     MY(SJ,0) = T0:DUMMY(SJ,1) =
     T1
50650  IF MODE$ = "Y" AND DUMMY(S
     J,1) < DUMMY(SI,1) THEN T0 =
     DUMMY(SI,0):T1 = DUMMY(SI,1)
     :DUMMY(SI,0) = DUMMY(SJ,0):D
     UMMY(SI,1) = DUMMY(SJ,1):DUM
     MY(SJ,0) = T0:DUMMY(SJ,1) =
     T1
50660  NEXT : NEXT : VTAB 23: HTAB
     1: CALL CL: RETURN
50700  REM
50701  REM  MEAN
50702  REM
50710 S1 = 0:S2 = 0: FOR LP = 1 TO
     NP(SET):S1 = S1 + D(SET,LP,0
     ):S2 = S2 + D(SET,LP,1): NEXT

50720 MX = S1 / NP(SET):MY = S2 /
     NP(SET)
50730  RETURN
50800  REM
50801  REM  ONERR GOTO ROUTINE
50802  REM
50810 EC =  PEEK (222):LN =  PEEK
     (218) +  PEEK (219) * 256
50820  IF EC = 5 AND CF THEN CF =
     0: GOTO 1000
50830 ER$ = "NUMBER " +  STR$ (EC
     ) + " IN LINE " +  STR$ (LN)
     : GOSUB 50000
50840  VTAB 23: HTAB 1: PRINT "RE
     SUME/CONTINUE/ABORT? (R/C/A)
     ";
50841 C =  PEEK (KBD): IF C < 128
     THEN 50841
50842  POKE KST,0:LC$ =  CHR$ (C -
     128)
50850  IF LC$ = "R" THEN  RESUME

50860  IF LC$ = "C" THEN  GOTO 10
     00
50870  PRINT D$"CLOSE";D$"PR#0": TEXT
     : HOME : PRINT "PROGRAM ABOR
     TED!": END
50998  REM
```

```
50999  REM  --------------------
       -
51000  REM  POINT
51010  HPLOT X,Y: RETURN
51100  REM  CIRCLE
51110  HPLOT X - 1,Y - 2 TO X + 1,Y - 2:
       HPLOT X - 1,Y + 2 TO X + 1,Y + 2
51120  HPLOT X - 2,Y - 1 TO X - 2
      ,Y + 1: HPLOT X + 2,Y - 1 TO
      X + 2,Y + 1
51130  RETURN
51200  REM  SQUARE
51210  HPLOT X - 2,Y - 2 TO X - 2
      ,Y + 2 TO X + 2,Y + 2 TO X +
      2,Y - 2 TO X - 2,Y - 2
51220  RETURN
51300  REM  DELTA
51310  HPLOT X - 2,Y + 2 TO X + 2
      ,Y + 2 TO X,Y - 2 TO X - 2,Y + 2
51320  RETURN
51400  REM  SOLID CIRCLE
51410  HPLOT X - 1,Y - 2 TO X + 1
      ,Y - 2: HPLOT X - 2,Y - 1 TO
      X + 2,Y - 1: HPLOT X - 2,Y TO
      X + 2,Y: HPLOT X - 2,Y + 1 TO
      X + 2,Y + 1: HPLOT X - 1,Y +
      2 TO X + 1,Y + 2
51420  RETURN
51500  REM  SOLID SQUARE
51510  FOR LY = Y - 2 TO Y + 2: HPLOT
      X - 2,LY TO X + 2,LY: NEXT
51520  RETURN
51600  REM  SOLID DELTA
51610  HPLOT X - 2,Y + 1 TO X - 2
      ,Y + 2: HPLOT X + 2,Y + 1 TO
      X + 2,Y + 2
51620  HPLOT X - 1,Y - 1 TO X - 1
      ,Y + 2: HPLOT X + 1,Y - 1 TO
      X + 1,Y + 2
51630  HPLOT X,Y - 2 TO X,Y + 2
51640  RETURN
```

All the subroutines are written in about the same way, using GETWORD (GOSUB 50200) to help parse their parameters, and then manipulating the global data structures or doing some other action. They all terminate with a RETURN to the main loop.

About the author. . . .

Tom Little is a student at New Mexico State University, working in Physics and Computer Science. He is currently the administrator and systems programmer for the Computer Science Department's UNIX system. He is also a contributing editor for Peelings II.

Tom has owned an Apple since 1979. His hobbies include writing science fiction (none published as of yet) and raising and hybridizing irises. Tom also enjoys doing artwork, mostly with pen-and-ink.

Become a "Millionaire" for $295.00

All **MILL** owners enjoy the "Millionaire" newsletter and the board that makes The Apple II/II+ faster than the III and The P.C.

The Lobero Building P.O. Box 2342
Santa Barbara, Ca. 93120
(805) 966-1140 Telex 658439

Last Minute Notes on DPCI

Our discussion of program development is basically concluded, but I'll add some notes on the program itself, in its final state, so that it may be easier to deal with or understand if you want to make use of it.

All the commands expecting a [set] will work on the active set as default, and otherwise expect a letter such as "A," "B," or "C" to indicate the proper set. If a set is specified this way, it becomes the active set.

Symbols are selected by their names ("POINT," "CIRCLE," "SQUARE", "DELTA"), and an optional adjective ("OPEN" or "SOLID"). "OPEN" is default. Thus, to plot with solid squares, one would respond to the "COMMAND:" prompt with "SYMBOL SOLID SQUARE".

"ECHO" defaults to ON. Only "ECHO OFF" will turn echoing off. It is initially off, however (line 310).

The default filename is "WORKFILE<DPCI>." The current filename is displayed on the screen, and will be used if "LOAD" or "SAVE" are entered without a filename parameter.

All words, of course, may be separated by either spaces or commas. Thus "LINEPLOT A, Y" works as well as "LINEPLOT A Y" or "LINEPLOT,A,Y".

"APPEND" puts you in APPEND MODE. In response to the "APPEND : " prompt, you may type in pairs of numbers indefinitely. (e.g. "123,4"). Exit by typing a carriage return only.

Some additional support subroutines were added. The one at 50500 refreshes the screen display. The display keeps you constantly informed of all the important system variables. An ONERR GOTO routine to trap Applesoft errors was added (50800). It allows you to either continue (jump to the main control loop), resume (at wherever the error occurred), or abort the program. It prints out the number of the Applesoft error, so you'll need your manual nearby.

The program is too large to fit below Hi-Res page one, so an EXEC file is used to load it above the graphics area (listing 2). This wastes memory that might otherwise be used for data. If the room is needed, the program might be split with a split utility.

Listing 2

```
]LIST

10 D$ =  CHR$ (4)
20  PRINT D$"OPEN START
      DPCI"
30  PRINT D$"WRITE START
      DPCI"
40  PRINT "POKE 16384,0"
50  PRINT "POKE 103,1"
60  PRINT "POKE 104,64"
70  PRINT "POKE 175,1"
80  PRINT "POKE 176,64"
90  PRINT "RUN DPCI"
100  PRINT D$"CLOSE"
```

It should be easy to make modifications to this program. The size and number of the data sets can be altered by changing line 130. No other modification should be required. New commands may be added by writing a subroutine, putting its line number in the GOSUB statement on line 1060, its name in the DATA statement on line 222, and incrementing the value of NC (number of commands) on line 220. Areas for expansion might be labels and axes. Any takers?

Listing 3

```
DOS MON I
TEXT
DOS CATALOG
GRAPHICS

ECHO ON
APPEND
   45,60
   50,76
   83,20
   90,102
   100,54
   35,30
   102,73
   115,90
   74,83
   60,41

AUTOSCALE
SYMBOL CIRCLE
PLOT

CLEAR
SYMBOL SOLID DELTA
ORIGIN 30,0
SCALE 2,1.4
LINEPLOT
CLEAR
LINEPLOT A Y
BARGRAPH A Y
CLEAR
BARGRAPH
LOAD MYDATA
AUTOSCALE
PLOT A
SYMBOL SQUARE
PLOT B

DOS NOMON I
```

One last tidbit: the stored program idea didn't die a lonely death. DPCI has the option of taking its input from a text file (listing 3 is an example of such a "command file"). This is like an EXEC, but a little more controlled. Being an interpreter, it doesn't really matter where the commands come from. Canned presentations can be made this way, and it seems to work quite elegantly. When DPCI starts up, it asks whether or not to run interactively. If you answer no, it will take input from the text file you name. Try to do this with a top-down, menu-driven program!

Conclusions

This program took only a few days to write and was quite rewarding. I attribute this ease to the bottom-up methodology that was used.

Obviously there are two ways to regard a computer program. One is as a solution to a problem (the top-down view). The other is as a set of tools with which the user may solve his own problems (the bottom-up approach). Obviously, the latter requires more understanding from the user, because the program does not decide for him what must be done.

There are also many times when you don't want a set of tools. But when you do, it's nice to be able to create them. Most fancy software (compilers, language systems, word processors, and so on) are designed in a manner much like that described here. There really isn't a great deal of difference, in principle, between writing a Data Plot Command Interpreter and a text editor or programming language. Such a project is not out of reach, if you approach it one primitive at a time, from the bottom up.

Apple® Business Software: ACCOUNTING PLUS II® from $329

Now, Pay Less, and get Great Service, too.

Finest, easiest-to-use accounting system for Apple/Franklin Computers. Now, at a special INTRODUCTORY DISCOUNT PRICE.

Accounting Plus II is recognized for completeness and efficiency. Speed, too, because it uses 6502 machine language. Just push a button and start. Comes with its own firmware card, no extra hardware to buy – unlike many systems which require additional memory and other expensive cards. Upgradable, it grows with your needs – supports 2 or 3 floppy drives or a hard disk. It works with virtually all printers. And it's also fully integrated, so all your entries automatically update all other applicable areas. (Easy-to-use manuals are included.)

24-Hour Customer Support. Price includes our complete customer support package. With your order, you'll be assigned a site identifier number and full membership in our special users group. Our technical support staff is on call 24-hours via toll-free WATS lines.

Send for complete package describing all the benefits of our *unique* products/Customer Support System.

Order by check, Visa or MasterCard. Fla. residents add 5% sales tax. Write or call: **1-800-327-7701. In Florida call collect: (305) 757-5416.** Delivery within 2 weeks. 30 day money-back guarantee.

Software Management Group, Inc.
12555 Biscayne Blvd., Dept. AP, Suite 805, Miami, Fla 33181

Systems Available:						Enhancements Available:			
Individual Modules	$299	G/L, A/P	$629	G/L, A/P, Inventory	$829	Data Plus	$ 99	Invoices Plus	$299
General Ledger	$329	G/L, Inventory	$629	G/L, A/R, Inventory	$829	Labels Plus	$ 99	Payroll	$329
G/L, A/R	$629	G/L, A/P, A/R	$829	G/L, A/R, A/P, Inventory	$995	G/L Plus	$199		

*Apple and Accounting Plus II are trademarks of Apple Computer, Inc. and Software Dimensions, Inc., respectively.

IMAGINE IT...

NEW

NOW WITH
DUAL HI-RES
GRAPHICS

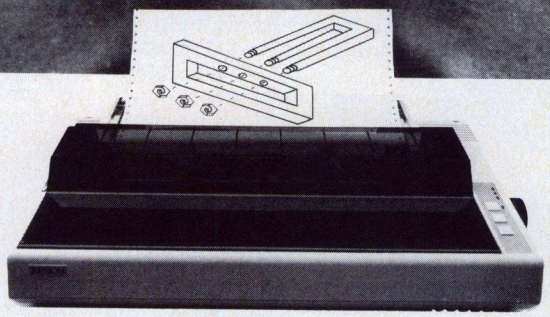
CAPTURE IT.

Completely Redesigned.
Now, the Grappler + .
The original Grappler was the first graphics interface to give you hi-res screen dumps from your keyboard. The new Grappler + with *Dual Hi-Res Graphics* adds flexibility with a side-by-side graphics printout of page 1 and page 2.

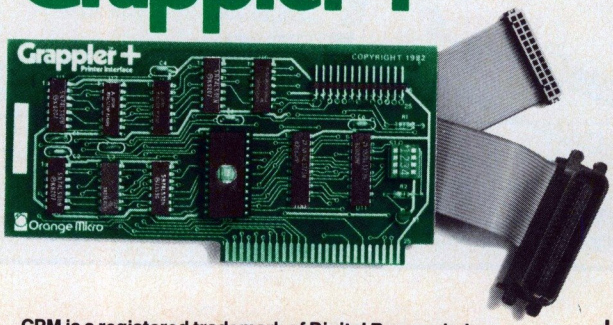
The Grappler + can now be used with the Apple® Dot Matrix, the Okidata 84, and is Apple III compatible.* In addition, the IDS Grappler + is currently available with color capability, including color graphics screen dumps.

The imitations are many, so insist on the #1 Apple graphics interface on the market. Insist on the Grappler + . Available now at most Apple dealers.

*Requires additional software driver.
**Requires graphics upgrade.



Grappler™ +



The Grappler + Features:
• Dual Hi-Res Graphics • Printer Selector Dip Switch • Apple III Compatible* • Graphics Screen Dump • Inverse Graphics • Emphasized Graphics • Double Size Picture • 90° Rotation • Center Graphics • Chart Recorder Mode • Block Graphics • Bell Control • Skip-over-perf • Left and Right Margins • Variable Line Length • Text Screen Dumps.
The Grappler + also works with Pascal and CPM.

The Grappler + interfaces with the following printers:
• Anadex • Apple Dot Matrix • Centronics 122 • C. Itoh ProWriter • Epson MX-70, MX-80**, MX-80F/T**, MX-100 • IDS 460, 560, Prism 80 and 132, Microprism • NEC 8023 • Okidata 82A**, 83A**, 84.

Orange Micro Inc.
1400 N. Lakeview, Anaheim, CA 92807
U.S.A. (714) 779-2772 Telex: 183511 CSMA
Foreign Dealer Inquiries Welcome

© Orange Micro, Inc. 1982

CPM is a registered trademark of Digital Research, Inc.
Apple is a registered trademark of Apple Computer, Inc.

A **Call -A.P.P.L.E.** TECHNOTE:

RND is Fatally Flawed

David Sparks/Call —A.P.P.L.E. Staff Writer

MUCH has been made of the erratic behavior of the Applesoft RND function. Tom Crawford in February 1981 described the problem: after a few thousand calls, the RND function starts repeating itself. Hardly a random number generator! Dave Lingwood discovered that the initial random number "seed" value provided by Applesoft is not always the same, and described his findings in *Call -A.P.P.L.E. in Depth #1.* But even using the repairs each described, the problem persisted.

This writer came forward with what he believed to be a random-number generator for use by assembly-language programs in the May 1982 issue. Though it passed all the tests we could think of, little time passed before a test appeared that we all flunked. Mssrs. Hare, Russ and Faulkner, whose article appears elsewhere in this issue, proposed something so simple that it hurt.

Their test calls for the generation of a series of number pairs such that the pairs describe plottable points on the Hi-Res screen. For example:

```
10   HGR : HCOLOR = 3
20   HPLOT RND(1)*280, RND(1)*
     160 : GOTO 20
```

will test the Applesoft RND function. Watch for a few minutes. When no more dots appear on the screen, it means RND has begun repeating exactly the same series of not-so-"random" numbers.

This will happen eventually regardless of the "seed" value originally provided. Crawford found that seeding with RND(-1) as described on page 102 of the Applesoft manual produces an especially long sequence; he tested it to 10 million values without finding a value which he described as being evidence of a repeating sequence. Unfortunately, the simple Hi-Res test above displays the fact of a repeating sequence long before the screen is filled with dots.

Dave Lingwood wrote up the random number seed value, as Applesoft stores it on zero page in bytes $C9 through $CD. He showed that byte $CD isn't always initialized to the same value. Although his article did not present the correct value for the offending byte, it led to the discovery of a "bug" in Applesoft. Yes, even systems programmer can make mistakes.

The bug causes Applesoft to properly initialize only the first four bytes of the random number field. If you wish to see the bug for yourself, get Applesoft up on your computer then enter the monitor by typing CALL -151. Type E000L [return]. You will get a screenful of "disassembly." You are interested in the first line, which says E000- 4C 28 F1 JMP $F128. This is the very first instruction in Applesoft, and is executed whenever you turn on the machine, type [Ctrl-B] from the monitor (assuming you have Applesoft on the motherboard ROMs), or issue the FP command of DOS. It causes the computer to "GOTO" location $F128 for further initialization instructions.

Meanwhile, back on your computer, type F128L [return]. You see the first part of the cold start initialization routine. Nothing here has anything to do with the problem, so type another L (with the usual [return] that you've come to always hit after typing any of this crazy stuff.) Look at the "disassembly" of line $F150. It says LDX #$1C. That is the bug. That sets up a loop that picks up data from within Applesoft ROMs and deposits it down lower in memory where Applesoft can manipulate it. The data includes the initial "seed" value for the random number generator. If the instruction read "LDX #$1D" then it would correctly get one additional byte, the essential final one of the "seed," and would store that byte where it belonged. But because of the bug, the last part of the "seed," byte $CD, does not receive any known value which can be consistently relied upon. Computer experts call that an "undefined" value.

What is the missing information? Return to your keyboard once more and type F123.F127 [return]. There you see the "seed" which comes tucked away in the very core of your Apple. For those who are not at your computer right now, the bytes displayed are: 80 4F C7 52 58. They are the "floating point" representation of the decimal fraction .811635157. That is the value the writers of Applesoft meant for the random number function RND to have. It is the final byte, 58, which does not get put where it belongs in location $CD.

So we put it there. We entered the monitor and put all five bytes where they belonged by typing CALL -151 [return], C9:80 4F C7 52 58 [return, 3D0G [return]. Then we ran the simple Hi-Res test, above. Failure. Dots soon stopped appearing; a repeating sequence had been found.

Here at last is conclusive evidence that the RND function is fatally flawed. It cannot be relied upon as a source of fair, properly distributed random numbers. If you write a gambling program using RND and your friends win all your money playing against your program, this is the reason why.

Hare/Russ/Faulkner call RND an "unexpected number generator", to distinguish it from algorithms known for more random results. But we have found instances of seeds which lead to repeating sequences only a few dozen values in length. Such a sequence would not even produce unexpected results very long.

RND IS FATALLY FLAWED

They refer in their article to "arbitrary bit meddling with numbers to produce unexpected but not necessarily random results." We decided to follow the RND function through its course, to see if it were guilty of "bit meddling." It is.

The function is found at $EFAE in the Applesoft ROMs. Get into the monitor from Applesoft again and type EFAEL [return]. Here is a guided tour:

EFAE - EFB2 test the sign of the value between the parentheses which accompanied RND. If the value is negative, a branch is taken to NEWSEED at $EFCC.

EFB4 - EFB8 set up for and call a routine to move the existing seed to the main math register, which is called the Floating point ACcumulator (FAC).

EFBB - EFBC test for a zero in the parentheses of RND. If found, a branch is taken to the RTS (return) instruction at EFA5.

EFBE - EFC2 set up for and call a routine to multiply the seed in FAC by a constant number which is kept in five bytes starting at EFA6. Those five bytes are 98 35 44 7A 68, and they represent the decimal value 11879546.4.

EFC5 - EFC9 set up for and call a routine to add the identical constant to the product of the multiplication just completed.

EFCC - EFD2 are the start of the NEWSEED routine. Here is the smoking gun. The four instructions simply exchange two bytes within the random number seed.

EFD4 - EFD6 force the sign of the number in FAC to be positive.
(You may need to type L [return] to see the rest)

EFD8 - EFDE set up some other values for...

EFE0 which calls a routine to "normalize" the new seed.

EFE3 - EFE7 exit the RND routine by jumping to a routine which saves the new random number as the seed for the next one. The RTS of the routine JMPed to returns to Applesoft.

That's it. RND does indeed indulge in bit meddling. So did our little effort, RNDGEN (*Call -A.P.P.L.E. May 1982).* Both fail the simple Hi-Res test, above.

Hare/Russ/Faulkner are right. We need a new RND. And thanks to them for finally presenting an answer to the random number problem. We can finally lay that one to rest.

About the author. . . .

David Sparks, 33, is a stockbroker in Waterloo Iowa. He bought his Apple II in 1980, brought it home, and set about to learn all he could on his own. He does not use the computer for picking investments, but does perform some client record-keeping, and expects to make use of the Apple in financial planning.

He encountered Call -A.P.P.L.E. *in 1981, when he began trying to learn machine language. The projects he undertook lead to several articles in these pages. He told us, "Publishing in* Call -A.P.P.L.E. *could easily have been a lifelong dream. It is a surprising honor to see that dream fulfilled in less than a year."*

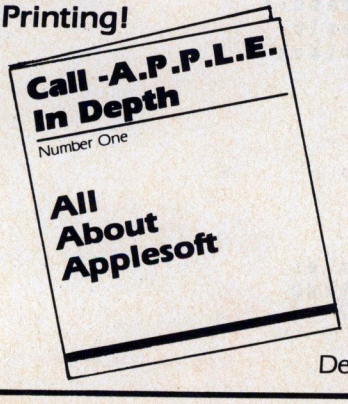
```
:ASM
      ************************************
2     *                                  *
3     *              USRND               *
4     *                                  *
5     *                by                *
6     *    Tom Hare, John Russ, Gary     *
7     *   Faulkner, and David Sparks     *
8     *                                  *
9     *   A random-number generator      *
10    *  for 48k ROM or L/C Applesoft    *
11    *                                  *
12    *  Call -A.P.P.L.E. : Jan. 1983    *
13    *                                  *
      ************************************
15    *
16
17    *
18    * The following machine language program is
19    * relocatable, but is shown here assembled
20    * to reside at $300. It is "hooked" into the
21    * Applesoft USR function when it is BRUN
22    * The calling format is (eg.) X = USR (A)
23    * if A is 1 the value will be a real number
24    * between 0 and 1 (the usual form for random
25    * numbers). If A is a positive number greater
26    * than 1, an integer 0...(A-1) is returned.
27    * Arguments between 0 and 1 always return 0, and
28    * arguments < 0 are rejected (error message).
29    * RND (0) returns the previous value.
30    *
31    * The algorithm for the random number generator
32    * starts with a seed value S and upon each
33    * call calculates a new seed value
34    * S' = (8192 * S) MOD 67099547
35    * Each random number is taken as the seed
36    * value divided by 67099547. This method
37    * will pass even quite exacting tests for
38    * "Randomness", and is strongly recommended
```

```
                        39   * as an alternative to the Applesoft RND function
                        40   * The original algorithm, which is used in some
                        41   * large computer programs, was introduced to
                        42   * us by Gary Faulkner, Dept. of Physical and
                        43   * Mathematical Sciences, North Carolina State
                        44   * University. It was first programmed in BASIC
                        45   * and subsequently in machine language by Tom
                        46   * Hare, School of Engineering, NCSU. His program
                        47   * was refined further by John Russ (Engineering
                        48   * Research, NCSU) and David Sparks (Call-APPLE
                        49   * staff writer). The accompanying article describes
                        50   * the method and several tests of this and other
                        51   * random number generators.
                        52   *
                        53            ORG   $2F4          ;A convenient location
                        54   *
02F4: A9 4C             55            LDA   #$4C          ;"JMP"
02F6: 85 0A             56            STA   $0A           ;The USR vector
02F8: A9 0F             57            LDA   #<RANDOM      ;Program address
02FA: 85 0B             58            STA   $0B
02FC: A9 03             59            LDA   #>RANDOM      ;Hi byte too
02FE: 85 0C             60            STA   $0C
0300: A0 04             61            LDY   #4            ;Use the 4 bytes from
0302: B9 C9 00          62   INITR     LDA   $C9,Y         ;Applesoft's RND seed
0305: 39 BC 03          63            AND   MASK-1,Y      ;to create our seed
0308: 99 B8 03          64            STA   BYT1-1,Y      ;(which may not exceed
030B: 88                65            DEY                 ;the magic 67099547)
                        66   * NOTE: this means that the random number
                        67   * generator can be seeded by RND (-N) and CALL 768
030C: D0 F4             68            BNE   INITR
030E: 60                69            RTS
                        70   *
030F: A5 9D             71   RANDOM    LDA   $9D           ;check for 0
0311: F0 50             72            BEQ   DIVIDE        ;return last one
0313: A5 A2             73            LDA   $A2           ;check positive
0315: 10 03             74            BPL   SAVE
0317: 4C 99 E1          75            JMP   $E199         ;"ILLEGAL QUANTITY"
                        76   *
031A: A0 03             77   SAVE      LDY   #>RANGE       ;save the argument
031C: A2 C1             78            LDX   #<RANGE       ;using the Applesoft
031E: 20 2B EB          79            JSR   $EB2B         ;move FAC->mem. subr.
0321: A2 F3             80            LDX   #$F3          ;-13 into counter
0323: 0E B9 03          81   SUBLOOP   ASL   BYT1          ;shift the seed 13
0326: 2E BA 03          82            ROL   BYT2          ;times to multiply
0329: 2E BB 03          83            ROL   BYT3          ;by 8192
032C: 2E BC 03          84            ROL   BYT4
032F: 38                85            SEC                 ;and try to subtract
0330: AD B9 03          86            LDA   BYT1          ;67099547 to perform
0333: E9 9B             87            SBC   #155          ;the MOD function
0335: 48                88            PHA                 ;NOTE- the values
0336: AD BA 03          89            LDA   BYT2          ;shown are the floating
0339: E9 DB             90            SBC   #219          ;point representation
033B: 48                91            PHA                 ;of the number. The
033C: AD BB 03          92            LDA   BYT3          ;difference is saved
033F: E9 FF             93            SBC   #255          ;on the stack for now
0341: 48                94            PHA
0342: AD BC 03          95            LDA   BYT4
0345: E9 03             96            SBC   #3
0347: 48                97            PHA
0348: 90 12             98            BCC   LESS          ;branch if < 67099547
034A: 68                99            PLA                 ;otherwise replace
034B: 8D BC 03         100            STA   BYT4          ;the seed value
034E: 68               101            PLA                 ;with the MOD result
034F: 8D BB 03         102            STA   BYT3          ;which we cleverly
0352: 68               103            PLA                 ;saved on the stack
0353: 8D BA 03         104            STA   BYT2          ;while testing
0356: 68               105            PLA
0357: 8D B9 03         106            STA   BYT1
035A: B0 04            107            BCS   INCR          ;forced branch
035C: 68               108   LESS      PLA                 ;gotta clean up
035D: 68               109            PLA                 ;the stack!
035E: 68               110            PLA
035F: 68               111            PLA
0360: E8               112   INCR      INX                 ;increment counter
0361: D0 C0            113            BNE   SUBLOOP       ;and keep shifting
0363: A9 03            114   DIVIDE    LDA   #3            ;place 67099547
0365: 85 9E            115            STA   $9E           ;into FAC
0367: A9 FF            116            LDA   #255
0369: 85 9F            117            STA   $9F
```

RND IS FATALLY FLAWED

```
036B: A9 DB      118            LDA  #219
036D: 85 A0      119            STA  $A0
036F: A9 9B      120            LDA  #155
0371: 85 A1      121            STA  $A1
0373: A9 00      122            LDA  #0
0375: 85 A2      123            STA  $A2
0377: 85 AA      124            STA  $AA
0379: 85 AB      125            STA  $AB
037B: A9 81      126            LDA  #$81
037D: 85 A5      127            STA  $A5
037F: 85 9D      128            STA  $9D
0381: AD BC 03   129            LDA  BYT4        ;and move seed value
0384: 85 A6      130            STA  $A6         ;to auxiliary FAC
0386: AD BB 03   131            LDA  BYT3        ;before multiplying
0389: 85 A7      132            STA  $A7
038B: AD BA 03   133            LDA  BYT2
038E: 85 A8      134            STA  $A8
0390: AD B9 03   135            LDA  BYT1
0393: 85 A9      136            STA  $A9
0395: A9 9D      137            LDA  #$9D
0397: 20 2E E8   138            JSR  $E82E       ;Normalize FAC
039A: 20 69 EA   139            JSR  $EA69       ;Divide AUX/FAC
                 140  * These Applesoft calls leave the random number
                 141  * in the floating accumulator (FAC)
039D: AD C1 03   142            LDA  RANGE       ;now check the user's
03A0: 38         143            SEC              ;RANGE argument
03A1: E9 81      144            SBC  #$81        ;is it a 1 (exactly)
03A3: 0D C2 03   145            ORA  RANGE+1
03A6: 0D C3 03   146            ORA  RANGE+2
03A9: 0D C4 03   147            ORA  RANGE+3
03AC: D0 01      148            BNE  SCALE       ;NO
03AE: 60         149            RTS              ;YES (return with fraction)
03AF: A0 03      150  SCALE      LDY  #>RANGE     ;load address to use
03B1: A9 C1      151            LDA  #<RANGE     ;Applesoft's
03B3: 20 7F E9   152            JSR  $E97F       ;MULTIPLY subroutine
03B6: 4C 23 EC   153            JMP  $EC23       ;then take integer
                 154  * part and return to calling program
                 155  * by using Applesoft INTEGER subroutine.
03B9: 00         156  BYT1       DFB  $0          ;4 safe bytes to
03BA: 00         157  BYT2       DFB  $0          ;hold the seed
03BB: 00         158  BYT3       DFB  $0
03BC: 00         159  BYT4       DFB  $0
03BD: 9A DB FF
03C0: 03         160  MASK       DFB  154,219,255,3 ;mask to assure
                 161  * that the seed value is < 67099547.
03C1: 00 00 00
03C4: 00 00      162  RANGE      DFB  0,0,0,0,0   * 5 bytes for range

--End assembly--

210 bytes

Errors: 0
```


Now Available!
2nd Printing!



IN DEPTH NO. I ALL ABOUT APPLESOFT

136 pages of timeless reference material.

- Applesoft from Bottom to Top
- Applesoft Internals
- Program Splitter
- Amplifying Applesoft
- Hi Res Notes

And More!

Member Price $7.00
Non-member price $7.50
(Available at your local computer store)

Dealer Inquires Invited

A.P.P.L.E. Orders
304 Main Ave. S.
Renton, WA 98055

Foreign members add $6.00 shipping/handling
Washington state resident add 7.0% sales tax
Canadian members, postage being revised

A NEW PSEUDO— RANDOM NUMBER GENERATOR

Tom Hare/John Russ/Gary Faulkner

BY now, most serious users of the Apple for simulation purposes, or even in many cases for games, have become aware of the inadequacy of the random number generator provided by the RND function in Applesoft BASIC. The sequence of values provided can fall into one of several rather short (as little as 200) repetitive loops, and even with reseeding the values provided will not pass even simple tests of having a random nature. It is not our purpose here to define rigorously what a random number really is, but for proper use in many simulation programs, not only must the numbers themselves be uniformly distributed over the interval from 0 to 1, but also the differences between successive values (and by extension, between any ordered pair) must also be random. Viewed in these terms, the Applesoft RND function is not really a random number generator at all, and we prefer to call it an "unexpected number generator".

One well known technique for generating psuedo random sequences of numbers, commonly used in large machines, is to start with some seed value and them repeatedly multiply it by one large number and take the remainder after dividing by another large number. The choice of the numbers is more an art than a science, and several pairs of useful numbers have been reported and widely used. The two numbers used here are relatively prime, and the length of the loop before repetition occurs is over 67 million. This particular pair is 8192 and 67099547; the new seed value is derived from the old one as:

S' = (8192 * S) MOD 67099547

and each random number is taken as the seed value divided by 67099547.

The machine language program shown accompaning David Spark's article carries out this operation and returns a random value into a named variable, using the USR vector into Applesoft.

When it is BRUN, the routine is installed from $300, $3C5 and the seed value initialized using the Applesoft RND function. It can be re-initialized by CALL 768 to repeat a series of numbers, and the seed changed by using RND(-N) and then CALL 768.

To test this routine, and demonstrate its superiority to the Apple RND routine, you can simply generate coordinates and plot points on the Hi-Res display, for instance:

```
10 HGR: HCOLOR = 3:
   POKE 49234,0
20 X=USR(280): Y=USR(192)
30 HPLOT X,Y: GOTO 20
```

The screen will eventually be completely filled with dots, whereas if X=RND(1): Y=RND(1) is substituted in line 20, it will not. A more discriminating test, which is very sensitive to the sequence of values (and also more fun) is presented here in the form of a simple simulation, which we call the "License Plate Game" (anyone who has travelled by car with small children probably already know it well).

The purpose of the game is to construct an alphabet from letters in the license plates of cars. We will presume that all 26 letters are used, and that there are three letters per plate. You must find an A before a B, and so on, and only one letter can be taken from a given plate. How many cars do you have to examine to obtain a complete alphabet? The program below carries out the simulation, and also displays the distribution of results. Figure 1 shows the plot for 7500 alphabets, which required about 5 million random numbers. The distribution is reasonable and shows, within statistics, a smooth peak (the maximum value in the graph is 81 occurrences) forming with a most probable value of about 210 cars per alphabet.

In contrast, Figure 2 shows the result from 2900 alphabets using the Apple RND function. Although the vertical scale on the graph is 151 occurrences, there are many gaps in the plot. Most number of cars never occur, and others are repeated

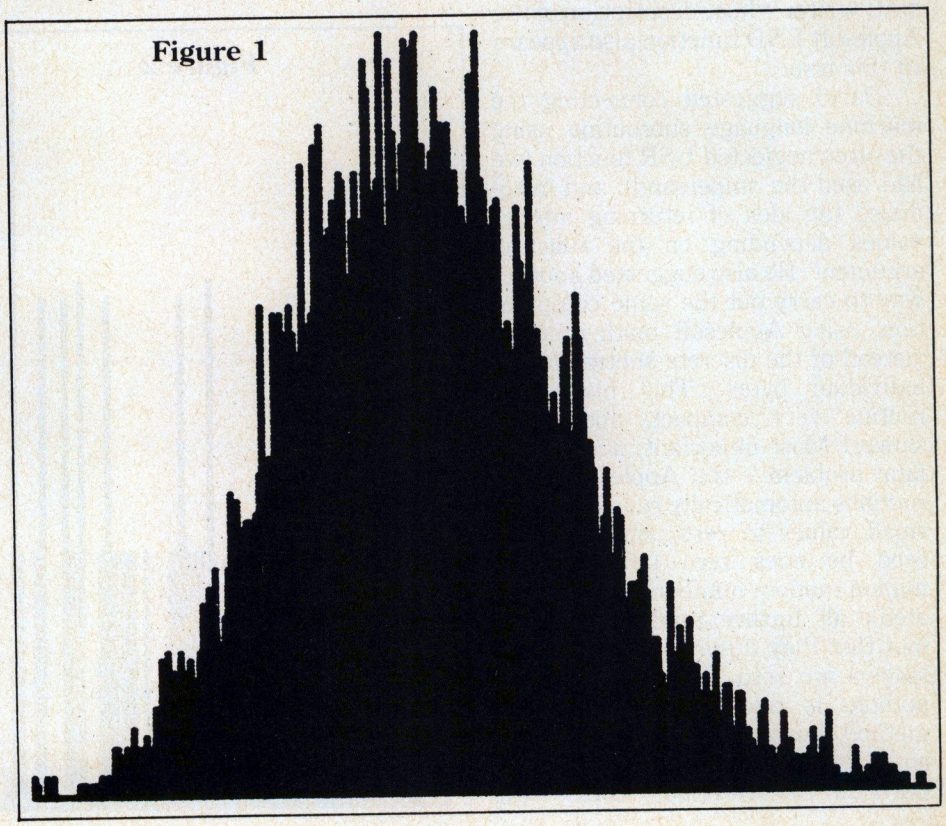


Figure 1

regularly (shown here with either about 50 or about 100 events). This is evidence of a repetitive sequence of values in the RND function output.

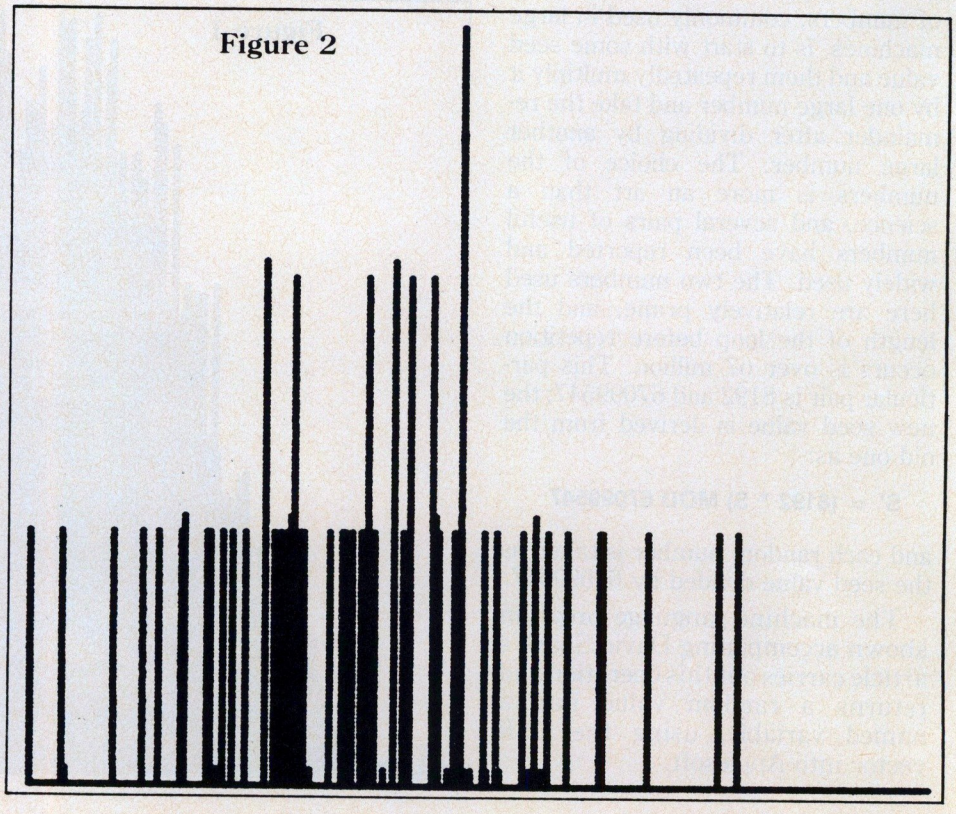
The new random number generator for the Apple is a proven reliable method, whose behavior is well known from use on large machines, and is therefore preferable to arbitrary bit meddling with numbers to produce unexpected but not necessarily random results. Also, the routine is easy to use and actually faster than the built-in RND function. We recommend its use especially for computer simulation studies with the Apple.

A brief description of how this program evolved may prove interesting. Originally, Gary Faulkner, in the N.C.S.U School of Physical and Mathematical Sciences, pointed out the basic method and provided the particular pair of mutually prime numbers. Tom Hare, in the School of Engineering, initially programmed the method in BASIC and subsequently in machine language. Then I modified his program to speed it up and correct some errors. At this point it seemed worth sending in to Call —A.P.P.L.E., where is ended up in the hands of David Sparks, a *Call —A.P.P.L.E.* staff writer whose description of the Applesoft RND function also appears in this issue.

David suggested connecting the machine language subroutine using the often neglected USR function (we had used the ampersand), and introduced the idea of returning integer values depending on the function argument. He also suggested another way to carry out the same computation, using Applesoft math routines instead of the discrete subtraction of individual bytes. This made the routine very compact, and a bit slower. Most important, it caused a fatal problem – the Applesoft math routines automatically round off very small values to zero, and once the seed becomes zero (after a few million random numbers were generated), all further random numbers (not that they deserve the name any longer) are zero. This falling into a pothole is avoided by using the method shown. David and I sent back and forth (from Raleigh, North Carolina to Waterloo, Iowa, with copies to Val Golding in Seattle, Washington) several versions, each improving on the other, until this joint effort was the outcome. We all truly believe that at last the RND problem can be laid to rest.

```
]LIST

5  REM

   LICENSE PLATE SIMULATION

JOHN RUSS : CALL -A.P.P.L.E.


10 N = 0: DIM NC(500):
     HOME : PRINT
     CHR$ (4)"BRUN
     USRND"
20 T = 0: FOR L = 0
     TO 25: REM T COUNTS
     NUMBER OF TRIES,
     L IS A LETTER
     (A...Z)
30   IF L =  USR (26)
     OR L =  USR (26)
     OR L =  USR (26)
     THEN 50: REM LOOK
     FOR 1 MATCH IN
     3 LETTERS
40 T = T + 1: GOTO 30:
     REM NO MATCH YET
50  NEXT L:NC(T) = NC(T)
     + 1:N = N + 1:
     PRINT N,T: REM
     KEEP TRACK OF
     RESULTS
60  IF  PEEK (49152)
     (  ) 141 THEN
     20: REM SEE IF
     USER PRESSES RETURN
     FOR DISTRIB.
70  POKE 49168,0:M =
     0:F = 1: FOR J
     = 500 TO 1 STEP
      - 1: IF NC(J)
     ) M THEN M = NC(J)
80  IF NC(J) ) 0 THEN
     F = J: REM FIND
     START OF PLOT
     AND MAX VALUE
90  NEXT J: HGR : POKE
     49234,0: HCOLOR=
     7: REM DRAW IT
100  FOR J = F TO 279
     + F:V = 191 *
     (1 - NC(J) / M):
     HPLOT J - F,V
     TO J - F,191:
     NEXT J
110  POKE 49168,0: INPUT
     "CONTINUE ";KB$:
     TEXT : GOTO 20:
     REM RESUME COMPUTING
```

Figure 2



Laing Electronics

10382 STANFORD AVENUE, SUITE E-203
GARDEN GROVE, CALIFORNIA 92640
PHONE (714) 534-4216

WIZARD 16 K CARD

By Wisper Micro - 2 Yr. Warranty

$64.00 Plus Shipping

★ HARDWARE FOR THE APPLE II ★

Bold Type indicates a PRICE REDUCTION ! !	LIST	YOUR COST
AMDEK VIDEO 300 MONITOR	249.00	143.00
NEC 12'' HI-RES GREEN MONITOR	285.00	153.00
HAYES MICROMODEM II	379.00	263.00
HAYES SMARTMODEM (RS232)	279.00	197.00
NOVATION APPLE CAT II	389.00	**268.00**
MICRO-SCI APPLE COMPATIBLE DISK DRIVE W/O CONTROLLER	395.00	**259.00**
MICROSOFT Z-80 SOFTCARD CP/M	395.00	**219.00**
MICROSOFT 16K RAM CARD	195.00	69.00
NEC PC-8023A PRINTER	695.00	468.00
OKIDATA MICROLINE 83A PRINTER	995.00	688.00
VIDEX VIDEOTERM CARD - 80 COL.	345.00	228.00
WIZARD 80 COLUMN CARD	249.00	**149.00**

★ DISKS BY THE CASE (100 TO A CASE) ★

BASF 5¼'' SOFT SECTORED	400.00	237.00
DYSAN 5¼'' SOFT SECTORED	600.00	363.00
MAXELL 5¼'' SOFT SECTORED	650.00	303.00
VERBATIM DATALIFE 5¼'' SOFT SECTORED WITH HUB RING	450.00	253.00

★ SOFTWARE FOR THE APPLE II ★

WORD STAR (MICRO PRO CP/M)	495.00	299.00
SUPER SORT (MICRO PRO CP/M)	250.00	152.00
MAIL MERGER (MICRO PRO CP/M)	250.00	152.00
DATA STAR (MICRO PRO CP/M)	295.00	179.00
SPELL STAR (MICRO PRO CP/M)	250.00	152.00
CALC STAR (MICRO PRO CP/M)	200.00	99.00
FORTRAN-80 (MICROSOFT CP/M)	195.00	125.00
BASIC COMPILER (MICROSOFT CP/M)	395.00	252.00
A.L.D.S. (MICROSOFT CP/M)	125.00	80.00

★ PERSONAL SOFTWARE ★

VISIFILES	250.00	179.00
VISIPLOT	200.00	143.00
VISITREND/VISIPLOT	300.00	215.00
VISICALC 3.3	250.00	179.00

★ SOFTWARE PUBLISHING ★

PERSONAL FILING SYSTEM	125.00	76.00
PERSONAL REPORT SYSTEM	95.00	58.00

★ STONE WARE ★

D B MASTER	229.00	147.00
D B UTILITY PACK	99.00	64.00

ACCEPTED PAYMENT - PERSONAL CHECK or MONEY ORDER. NO CREDIT CARDS.
If Personal Check, Allow 10 BANKING DAYS for CLEARANCE.
Prices and specifications subject to change without notice.
Unless otherwise noted, ALL PRICES are PLUS SHIPPING.
SHIPPING & HANDLING — ADD $5.00 UPS SURFACE (DISK DRIVES $7.50 — MONITORS $10.00 — PRINTERS $15.00).
California residents add 6% Sales Tax.

The Pizza Program

Announcing the first dinner menu planning system. It will save you time and add new zest to your meals. It may even convince your wife buying an Apple* was a stroke of genius.

ENDS HO-HUM DINNERS

Are you tired of the same old thing for dinner? Would you like more variety in your evening meal? Is there something you'd rather have but don't get very often? The Pizza Program is designed just for you. It's a delightful new software package designed to end the dinner-blahs with computer generated menus. Here is how it works.

You review what you like from the pre-selected food groups in the system. Delete any foods you don't enjoy. Add anything new at any time. Then decide how often you like to eat certain items. For example. Don't like liver? Then eliminate it with a few simple keystrokes. Or, you can plan for it as seldom as once every 99 weeks or as often as daily.

Want to go out to your favorite restaurant? Enter the restaurant's name as a "Main Course." Now your computer will automatically remind you to go out to eat — and as often as you select. It will delete all other items from that meal except the name of the restaurant.

AUTOMATIC SHOPPING LIST

You get a new menu each week or for just a few days if you want. And, it generates a detailed shopping list, automatically. It can arrange each item on the list in sequence according to the aisles at your favorite store. Studies show a shopping list will discourage impulse buying and save you money.

Also, it generates a per serving calorie counter. This is easy to delete anytime you are not in a diet mood or want to celebrate for any reason. You never count calories unless you want to.

RANDOMLY DELICIOUS

Say goodby to boring meals. Your computer will remember variety is the spice of life. This system makes eating at home a pleasure again. Each menu is randomly generated from major food groups according to the specific criteria you select. The system is easy to learn and easy to operate. Yet it is a sophisticated piece of software which will prevent menu mix-ups.

It will add a new dimension to your home meals without increasing your food costs. It is rumored The Pizza Program may actually reduce the climbing divorce rate. Husbands now have something to look forward to for dinner. Wives think it is terrific because it saves time and effort. And, kids love anything computerized. This is a useful and practical application you'll appreciate day after day, week after week.

TRY IT FOR 30 DAYS WITHOUT RISK

This tested system is guaranteed to make your life easier and happier. Our home trial lets you actually use The Pizza Program for a full 30 days before you decide to keep it. Watch the fun and convenience it creates. Enjoy better meals and see how much time it saves. If you are not satisfied for any reason, return it within 1 month for a prompt and courteous refund. Your investment is just $34.50 plus $2.00 for shipping and handling. (California residents add 6½% sales tax.) Full documentation is included.

OUT OF THE RUT

One housewife's reaction to this program is typical. She wrote, "Before using your system I found myself getting into a rut of serving the same things over and over. The Pizza Program has changed all of this for me. We now have a wider variety of dinners and best of all I don't have to decide what they will be. If this was all it did, I'd be thrilled. But it isn't. The shopping list I receive along with my menus has been such a time saver. I quickly run through it and delete anything I feel I don't need and add something I might. I would have a hard time going back to doing my menus by hand."

This system requires an Apple II Plus* with 48K and 1 disk drive. We urge you to take advantage of our no-risk, 30 day home trial offer. To order call toll free and use your VISA or MasterCard. Or, send a check to the address below. There's no obligation. Order today.

ORDER TOLL FREE
(24/hrs/day)

800-453-4000
(In Utah call **1-800-662-8666**)

Gourmet Software

671 Eden Avenue
San Jose, CA 95117

*Apple is a registered trademark of Apple Computer, Inc.

FUTRA COMPANY

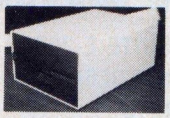
(213) 328-8951 (800) 421-5006
TWX 910 349 6211
AGENFTRA TRNC

Mail Order: **Department C**
P.O. Box 4380
Torrance, CA 90510-4380

Retail: **OMC** **20695 S. Western Ave.**
Suite 124
Torrance, CA 90501
(213) 328-1760

apple computer
Sales and Service

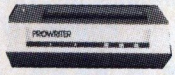
NEW FUTRA FLOPPY ADD-ON



The new Futra 5¼" Floppy Disk Drive is fully software transparent with DOS 3.3 and 3.2.1 operating systems, as well as Pascal and CP/M. Add it to your present drive and controller for greater capacity and faster access. Just plug it in and go to work.
The Futra Apple-beige unit is built around a highly reliable Siemens system with over 10,000 lifetime hours. A shielded connecting cable is attached.
A Futra drive makes an excellent investment in preventing down time. At this price you don't have to be down a week waiting for your Apple drive to be fixed. Keep a spare Futra Drive just in case.
The Futra Drive features On-Track Head Seek with a precision lead screw to position the head onto the correct track. Retries and disk-to-disk copy errors are virtually eliminated.
Futra Drive has a faster Track Access Time at 25 MS than Disk II (40 MS), and a faster Average Access Time at 298 MS (Disk II at 463 MS).
The unit has a one-year warranty and MTBF (Mean Time Between Failures) is 8500 power-on hours. That's almost 3 years based on 8 hours per day.

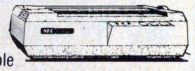
PRINTERS

OKIDATA
- Microline 82
- Microline 84



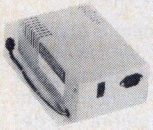
C. ITOH
- Prowriter
- Starwriter
- Printmaster

NEC 8023A



Grapplers Available

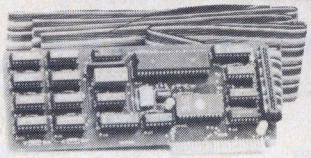
POWER



M & R Enterprises
SUP 'R' SWITCHER
$219.00
substitute supply

GE Surge Suppressor
Protect against power spikes ... **$7.50**

MICROTEK Apple Dumpling-64
Spooler/Buffer for Apple Computers

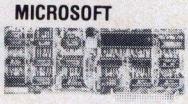


- Standard Dumpling GX **119.00**
- Dumpling-64 w/16K Buffer **179.00**
- Dumpling-64 w/32K Buffer **202.00**
- Dumpling-64 w/64K Buffer **225.00**

In addition to the standard graphics features, the Apple Dumpling-64 is the next logical extension to the graphic dump boards. Now you can get all the standard graphics features and up to 64K of buffer memory. Why tie up your computer waiting for your printer to catch up. Just dump the information into the buffer area and away you go. You're not just buying an interface you're buying time.

The Apple Dumpling-64 allows full use of hi-resolution functions:
Rotation, Inversion, Double Size, Emphasis Print, Dual Page Dump, Chart Recorder Mode and control over all variables for text and graphics.
Of course, there's PASCAL AND CP/M compatibility, ideal default conditions, and a standard 2 year warranty. Available for most graphics printers.

MICROSOFT

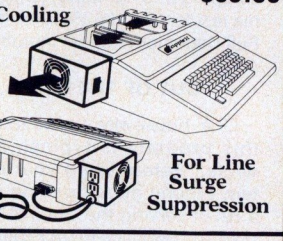


$ CALL $
Z-80 Soft Card with CP/M

$ CALL $
16K Ram Card

Futra brings you the industry standard at it's lowest price ever. When you can get the trusted Microsoft quality, why buy anything else?

KENSINGTON MICROWARE
System Saver Fan **$69.00**
For Cooling



For Line Surge Suppression

LPS II/PENTRAK SYSTEM from Gibson Laboratories
Light Pen System for Apple II

The LPS II is the only true High Resolution Light Pen System with full software support for the Apple II Computer. High Resolution pictures, diagrams and other graphics can be easily drawn on the screen of the Apple II. PENTRAK, the light pen driver, lets you easily create your own Applesoft programs for Light pen graphics. PENPAINTER is a color sketching system which allows the user to create an infinite variety of patterns to be used for filling-in the sketched areas. Area refilling allows various combinations of patterns to be tried and changed. Hi-Res Text generation is a standard feature of the PENTRAK driver, allowing simultaneous use of multiple user-defined character sets. List price $349.00. Call for Futra's special introductory price.

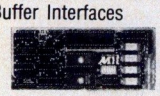
Possible Applications:
- Computer Aided Design
- Animation & Game Play
- Process Control
- Interactive Education
- Computer Aided Drafting
- Logic Design Simulation
- Business Graphics
- Text Editing
- Graphic Arts
- Menu selection
- Circuit Analysis

VIDEX INC.
VIDEOTERM
$ CALL $

When you're looking for a 80 column video board, why settle for anything less than Videx? Futra Co. makes it possible for you to own the best.

Manual Switch Plate	$19
Electronic Switch Plate	32.50
Micromodem II Firmware ROM	25.00

PRACTICAL PERIPHERALS
Buffer Interfaces



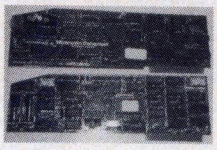
MICROBUFFER II INTERFACE

MICROBUFFER MBP-16K

Free your computer system from the binds of your printer!! Now you can get a buffer built into the interface that fits into the computer or a buffer that you install in the printer.

- Microbuffer II 16K **235.00**
- Microbuffer II 32K **279.00**
- MBS-8K Epson Serial (Printer) **149.00**
- MBS-16K Epson Parallel (Printer) **129.00**

MULTIFUNCTION CARDS

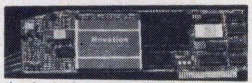


Mt. computer CPS Multi-function card	165.00
Prometheus Versa Card	159.00
Thunderware Thunderclock	105.00

HAYES MICROMODEM II $289

The Micromodem II is a complete communication system for the Apple II

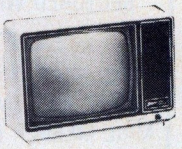
NOVATION APPLE CAT II $269



Buy the Apple Cat II, the modem that grows with your needs. Find out about Novation's add-on features

Video Terminals & Monitors

Zenith 12"
$CALL$

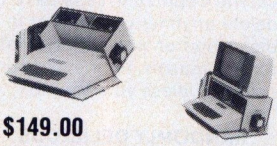


BMC 12"
NEC:
- 12" Green Monitor
- 12" Color Monitor (Composite)
- 12" Color Monitor (RGB)

APPLE SOFTWARE – GAMES

Automated Simulations		Time Zone	89.96
Temple of Apshai	29.00	Frogger	31.45
Crush, Crumble & Chomp	25.00	Soft Porn Adventure	23.96
Broderbund Software		**Sirius Software, Inc.**	
Apple Panic	23.96	Gamma Gobblins	26.95
Serpentine	31.50	Sneakers	23.96
Star Blazer	24.00	Escape from Rungistan	25.50
Midnight Magic	29.70	Kabul Spy	31.45
Arcade Machine	53.95	Bandits	27.96
Choplifter	32.50	Phantom Five	26.95
Budgeco		**Sir Tech**	
Raster Blaster	23.96	Wizardry	37.50
Infocom		Knight of Diamonds	27.96
Zork I	32.50	**Strategic Simulations**	
Zork II	32.50	Computer Quarterback	27.00
Deadline	44.95	Warp Factor	27.00
Muse Software		Cartels and Cutthroats	27.00
Three Mile Island	35.95	President Elect	27.00
Castle Wolfenstein	23.96	Computer Baseball	27.00
On-Line Systems		Tigers in the Snow	27.00
Jawbreaker	23.00	**Sub Logic**	
Threshold	34.50	Flight Simulator	26.80

COOL STACK – SENTRY II



$149.00

Protect your investment! In today's world of "Smash and Grab" it only takes a burglar a few seconds to make off with a computer that took you months or years to save for. Not only does the Cool Stack-Sentry II lock down, but it cools and organizes your system. Be safe not sorry.

ACRYLIC PLASTIC PRINTER STANDS

$22.50*

Put your printer on a tilted pedestal. Now you can check your printer with just a glance, and save space by storing your paper under the printer. Specify Epson MX-80, NEC 8023A, or Okidata Microline 82.

Epson MX-100 **$32.50***

*Available in Smoke Bronze

DATA SUPPLIES

Flip Sort
Plastic Storage case **$27.00**

Scotch Products: For reliable data recording use the best diskette for the money. With reinforced center!
Box of 10 - 5¼" storage carton ... **$27**
Scotch Head Cleaning Kit **$24**

MOUNTAIN COMPUTER

Apple Clock	219.00
ROM Plus	125.00
ROM Writer	141.00
CPS Multi-function Card	165.00
Music System	355.00
X/10 Control	160.00
Expansion Chassis	602.00

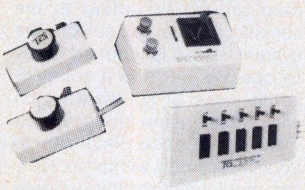
GENERAL APPLE SOFTWARE

Microsoft		**Visicorp**	
Cobol 80	$562	Visicalc	179
Fortran 80	175	Visitrend/Visiplot	239
ALDS	95	Visidex	211
TASC	149	Visiterm	125
Micropro		Visiplot	169
Wordstar	$235	Visischedule	253
Datastar	202	Visifile	211
Mail Merge	85	Visipack	549
Spellstar	129	**Stoneware**	
Calcstar	130	DB Master	$179
Supersort I	159	DB Master Utility Pack	$89

Continental Software
Home Accountant 65.00
Home Money Minder 29.70

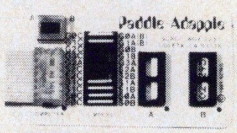
Software Publising Corp. PFS: Graph
Personal Filing Systems 87.50
Personal Report Systems 85.50 109.00

TG PRODUCTS



Joysticks	42.00
Game Paddles	27.50
Selectaport	42.00

PADDLE ADDAPPLE $27.00



Game I/O over loaded? Don't open your Apple. Get the two-plug Paddle-Addapple. Switch selectable and reasonably priced.

TERMS: Shipping: add 3% for product shipped within continental USA via ups surface (minimum $3.00). **If the order placed is prepaid with U.S. funds in the form of check or money order, a Total Charge of $3.00 for shipping is all you pay within the continental USA vis UPS surface.** Allow 14 working days for personal and company checks to clear. Credit card charges limited to $1000. No COD's. FPO, APO or orders outside continental USA call or write for shipping charges or add 20% to purchase price (any difference will be refunded). CA residents add 6% sales tax (LA County 6.5%). All offers subject to change or withdrawal without notice.

*CPM is a registered trademark of Digital Research.

MACHINE POWER:

WHERE TO PUT MACHINE CODE

Bill Parker/Call —A.P.P.L.E. Staff Writer

THIS is the first part of a trilogy of articles on how to get the most out of your Apple by using machine code utilities to add zip and power to your system.

These articles will show you how to collect, organize, place and use your machine language utilities in a way that will keep them from being overwritten by other programs and at the same time keep them from interfering with each other. You will not be taught the basics of assembly or machine code language, however... there are many excellent books on this topic out on the marketplace right now.

We'll begin by considering some good places to put machine code programs and the reasons why they should be put there. But first, some background.

By now, chances are pretty good that you are already intuitively aware of how machine code can greatly benefit your programming. Consider, for example, how these two benchmark programs perform the mundane task of clearing the screen 100 times:

```
10  FOR SCREEN = 1 TO 100
20  :FOR LINE = 1 TO 24
30  :: PRINT
40  :NEXT LINE
50  NEXT SCREEN
```

Program A
(no machine code)
Execution Time: 47 seconds

```
10  FOR SCREEN = 1 TO 100
20  CALL -936        *(machine
    code call)
30  NEXT SCREEN
```

Program B
(machine code call)
Execution Time: 2 seconds

If you actually run the first program, be prepared for a *loooong* wait; it will be nearly a minute before you see the cursor again.

Compare this to the second program, which is faster by a factor of 20! This is mainly due to the CALL -936 statement, which temporarily turns over program execution to a section of machine code in the monitor named, appropriately enough, "HOME".

What does the -936 mean? The number refers to a displacement from the top (if the number was positive it would refer to the bottom) of your Apple's memory:

64K = 64 X 1,024 = 65,536 total bytes in the Apple
- 936 bytes from the top
64,600 = $FC58 (in hex)

If you turn to page 147 of the Apple] [Reference Manual (and no good machine code user would be caught dead without one), you'll see near the top of the page the HOME routine at $FC58. *This is where your Apple goes when you say CALL -936.* Because Applesoft has relinquished control temporarily to this routine and can no longer check every step of the way, the system can attain an incredible increase in speed.

HOME is just one of many "entry points" into the monitor, which is itself just a collection of useful machine code subroutines. By setting conditions just right and calling these routines in the proper order, you can do all sorts of marvelous things not normally available from BASIC.

Here's an example of a simple, one line Applesoft program that converts decimal into hex using a monitor call:

```
10  INPUT "DEC = ";D
    DH% = D/256 : DL % = D-
    256*DH%:
    POKE 69,DH% : POKE 70,
    DL% : POKE 58,65 : POKE
    59,249 :
    POKE 49,0 : PRINT "HEX = $";
    : CALL -327
]   RUN
    DEC = 64600
    HEX = $FC58
```

While it is simpler to write and run the program than it is to try to understand how it works, nevertheless here is an explanation for the lion hearted:

1. Break the decimal number up into a "high-order" byte and a "low-order" byte (DH% and DL%).
2. Put the bytes in save areas reserved for the A register and the X register (locations 69 and 70).
3. Put the address of the monitor routine that prints the contents of the A and X registers (PRNTAX) into the pseudo-Program Counter (locations 58 and 59).
4. Clear MODE (location 49), which tells the monitor how to "parse" or dissect your number.
5. Call monitor routine REGZ (-327), which loads the registers from the save areas described in step 2 above and jumps to the address you specified in step 3.

(You can get more details on this and how to use other useful monitor routines in the book, *Apple] [Monitors Peeled,* published by Apple Computer.)

But why stop with monitor routines? Applesoft and DOS are also vast machine code programs; can we use routines within them as well? The answer is *yes* and computer magazines and newsletters are filled with examples of such use.

(Tip: Two good sources for information on Applesoft: A completely disassembled and commented source listing of Applesoft comes with Merlin, an assembler from Southwestern Data Systems, and one of the best articles showing how to use Applesoft routines is the hallmark article by John Crossley on pages 51 to 57 of *Call - A.P.P.L.E. In Depth — No. 1,* "Applesoft Internal Entry Points.")

Now, while there are literally thousands of routines floating about and there are just as many stories written showing how to use them, it is quite another matter to know how to get finger-tip access to them in just the order you need to augment a program you're developing. Which brings us back to the focus of this

—◉→

What's eating your Apple?

Find out with Apple-Cillin II™

If you use your Apple for your business or profession, you probably rely on it to save you time and money. You can't afford to guess whether it is working properly or not. Now you don't have to guess. Now you can find out with Apple-Cillin II.

Apple-Cillin II is the comprehensive diagnostic system developed by XPS to check the performance of your Apple II computer system. Apple-Cillin II contains 21 menu driven utilities including tests for RAM memory, ROM memory, Language Cards, Memory Cards, DISK system, Drive Speed, Keyboard, Printer, CPU, Peripherals, Tape Ports, Monitors and more. These tests will thoroughly test the operation of your Apple, and either identify a specific problem area or give your system a clean bill of health. You can even log the test results to your printer for a permanent record.

Apple-Cillin II works with any 48K Apple system equipped with one or more disk drives.

To order Apple-Cillin II - and to receive information about our other products - Call XPS Toll-Free: 1-800-233-7512. In Pennsylania: 1-717-243-5373.

Apple-Cillin II: $49.95. PA residents add 6% State Sales Tax.

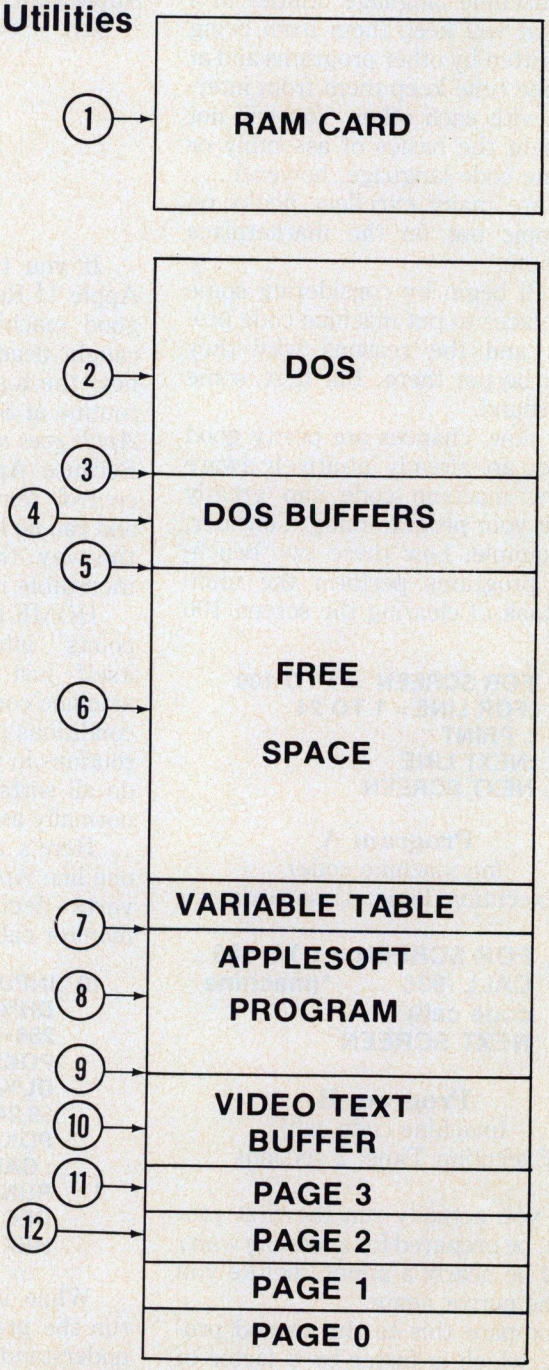
XPS

XPS, Inc.
323 York Road
Carlisle, Pennsylvania 17013
800-233-7512
717-243-5373

Apple II is a trademark of Apple Computer Inc

APPLE II SYSTEM MEMORY MAP

Valid Places To Put Machine Code Utilities



article: Where do you put all these machine code programs you find in magazines?

It is my contention that we need to pull our heads out of the sand, take a good look at all of the resources available to us and come up with an organized discipline to implement machine code utilities.

I believe the first thing we should all do is to set aside a disk or two just for storing all those nifty machine code utilities you find in magazines. Next, we need to know good places in memory to put these routines so that they will be safe from being overwritten and will not interfere with each other. Finally, we need to have a systematic method of bringing any combination of these routines into play just when they are needed.

The remainder of this article explores areas in memory to put machine code programs and the reasons why.

As it turns out, there are just 12 valid locations to put a machine code program in memory (see facing page).

1. On A RAM Card.

Method of implementation: By PEEKing or POKEing the range from 49280–49295 ($C080–C08F), you control which memory area in the range of $D000–$FFFF will be used, from the RAM card or from the "motherboard" ROM (i.e., the Apple). Exact details are available in the manual that comes with the card. *Advantages:* A large amount of memory is available. *Disadvantages:* Complexity of issuing the access codes within a running program. May conflict with other programs residing on the card (DOS, BASIC, GPLE, BigMac, etc.). *Good article: Call -A.P.P.L.E.* Sept. 82 page 43, "Adventures with a RAMCARD."

2. Within DOS.

Method of implementation: Overwrite existing code and pointers, such as replacing the "MAXFILES" command with a "FILEDUMP" command and changing the MAXFILES pointers to point toward an unused area of DOS where new code is inserted. For details on this, see the June issue of *Call -A.P.P.L.E.*. *Advantage:* Once installed, commands are handy and easy to use. *Disadvantages:* Disables portions of DOS; difficulty in keeping track of modifications in a DOS that has been altered several times; limited amount of space and commands available for alteration. *Good article: Call -A.P.P.L.E.,* July 82, page 49, "Patching DOS the Easy Way."

3. Between DOS and its buffers.

Method of implementation: The DOS buffer pointer at $9D00.9D01 must be checked to see where the DOS buffers begin. The machine code file is then loaded at that location and the buffers are rebuilt at the end of the machine code. *Advantages:* Plenty of room is available for long machine code files; most conflicts with other programs using this area are reduced, since the DOS buffer pointer keeps track of the next available loading address; unaffected by HIMEM:, MAXFILES, RESET or switching languages. *Disadvantages:* Unless you use an assembler or can load the code at the same address every time, the code must be relocatable. The DOS buffer pointer must be consulted first before loading files and buffers must be rebuilt beneath the file. An unusual procedure at first, but after a while becomes easy to remember and do. There are still some programs that blindly load files anywhere beneath DOS without regard to where the buffers are, thus possibly wiping out any files you may have stored there. *Good article:* Keep reading.

4. Within a DOS buffer.

Method of implementation: On a 48K RAM system, PEEK(43607) will tell you how many DOS buffers you have available. You can then set MAXFILES to one number greater than this value to get an extra (blank) buffer of 595 bytes in length! The beginning address of any buffer is found by the following formula:

```
PRINT PEEK (40192) +
256*PEEK(40193) -557-595* (N-1)
```

where N = the number of the buffer you are interested in (numbering starts at 1). You can then load your machine code file at that location. *Advantage:* Can hold large files. *Disadvantages:* Unless you use an assembler or are able to load the program in at the same location every time, the code must be relocatable. Space must be reserved in 595 byte blocks, which can get difficult to keep track of and also wastes space. Susceptible to attack from HIMEM: and MAXFILES.

5. Beneath the DOS buffers and above HIMEM:

Method of implementation: The current HIMEM: (top of available RAM) setting can be found with:

```
PRINT PEEK (115) + PEEK (116)*256
```

Load your machine code file at this address and reset HIMEM: to the original value minus the length of your file. Warning: This will wipe out the Applesoft string (from INPUT statements) storage area, so make sure you perform this process before encountering any INPUT statements. *Advantages:* Easy to implement; one of the most common methods of providing space in the early days of Appledom. Still commonly used. *Disadvantages:* Susceptible to attack by HIMEM: and MAXFILES.

6. Within free space.

Method of implementation: Simply load your file anywhere beneath HIMEM: and above any possible Applesoft program. The beginning of available space following an Applesoft program can be found by:

```
PRINT PEEK(109) + PEEK(110)*256
```

Advantages: Largest of all available memory sources (about 36K). Typically used by commercial programs written entirely in machine language. *Disadvantages:* Cannot be co-resident with an Applesoft program that uses INPUT statements, since strings from these statements are stored anywhere in free memory, thus exposing your file to the chance of being overwritten.

7. Between the variable table and the Appplesoft program.

Method of implementation: The machine code file is loaded at the end of the Applesoft program and the LOMEM: pointer is then set at the end of the machine code file. When the Applesoft program is run, variables are stored in memory following the machine code file. Note that LOMEM: does *not* move the Applesoft program; it moves the variable table. This method is also known as the "transparent method" since the machine code is permanently attached to the BASIC program (i.e., is SAVEd and LOADed along with the program, and moved up and down in memory when the program is edited) but is never seen when the program

—◉→

WHERE TO PUT MACHINE CODE

is LISTed. *Advantages:* A separate binary file is not needed (i.e., no time consuming BLOADs and BRUNs are needed) because the BASIC program carries the machine code file with it. *Disadvantages:* Unless you are absolutely sure the Applesoft program will not be edited or moved around in memory, the code appended to the program must be relocatable. At least one statement in the program is needed to compute the beginning address of the routine and usually this statement is at a fixed and sometimes awkward location. Renumber programs may either fail or destroy the pointers protecting the machine code. *Good article: Call -A.P.P.L.E.,* July 82, page 59, "Dynamic Loading and Linking".

8. Within the Applesoft program.

Method of implementation: A variety of techniques are used here, including storing machine code in REMark statements, DATA statements, first statements and last statements. *Advantages:* Same as #7 above. *Disadvantages:* Same as #7 above, although this method is slightly more cumbersome. Also, longer routines must be broken up and sprinkled throughout the program in what could become a very complicated linking method. *Good article: Call -A.P.P.L.E.,* July 82, page 51, "Imbedding Machine Code in Applesoft Program Files".

9. Beneath the Applesoft program.

Method of implementation: Either the program is moved up in memory by the use of a utility such as "&LOMEM:" *(Apple Orchard #1)* or else the pointers to the beginning of the program are adjusted upwards. *Advantages:* Because the freed space always begins at $800 (2048), machine code files loaded there can be non-relocatable (this makes code easier and more compact to write). A large amount of space is also available. Typically used by "higher text" type programs that write text on the Hi-Res screens, the Applesoft program is placed on top of the Hi-Res screens, which leaves about 6K for machine code at the beginning of memory. *Disadvantages:* Few programs that use this technique check to see if there is already a machine code file residing at the low end of memory, thus forcing you to run the risk of having your file overrun. *Good article: Call -A.P.P.L.E.,* Jan. 82, page 75, "Pointers on Pointers".

10. Within the video text buffer.

Method of implementation: Load the machine code file anywhere between 1024 and 2039 ($400-7F7). Note: there are a few non-display bytes (64) that must not be used. *Advantages:* You are almost guaranteed that no other program will use this area because of its novel (and slightly unorthodox) nature and its disadvantages (see below). This also means you can use non-relocatable code. *Disadvantages:* Anything you put in the video text buffer is immediately displayed on the screen! While your program will run just fine, it will cause a lot of confusing garbage to be displayed. There are certain bytes within the buffer that are used as "scratchpad" locations by interface cards that you have plugged into the back of your Apple. *These locations must not be altered!* This can play havoc when trying to load lengthy files into this area. *Good article: Call -A.P.P.L.E.,* Feb. 82, page 9, "Examining the Video Text Buffer", (by the author).

11. Within page 3.

Method of implementation: You can load a machine code file of up to 207 bytes in length in the "page 3" range of $300.3CF (768-975). (Anything higher than this in page 3 will overwrite the DOS vectors residing there.) *Advantages:* Easy to remember starting address and most magazine articles list their source code at $300. Can take a program of up to intermediate length. Programs can be non-relocatable. Ideal for transient (use once and discard) use or for quick-and-dirty development. *Disadvantages:* There are more programs that use this space than can fit into this area together. Certain commercial programs use this area also.

12. Within page 2.

Method of implementation: Page 2 is also known as the keyboard buffer because it is the memory area set aside to temporarily store the characters you type in at the keyboard. Your machine code file can be loaded into the keyboard buffer (ranges from $200.2FF (512-767)). *Advantages:* Can be used to make a program fit that is just a little too long for page 3. Can also be used when it is necessary to load the keyboard buffer directly and call a routine that scans the buffer. *Disadvantages:* Using the keyboard buffer to store a program is like playing with fire. *The Apple needs this area to store things you type in.* Unless you plan on using a routine that scans the beginning of the input buffer, it's best to use the lower end (numerically higher) of this area and to use it only out of dire necessity. *Good article: Call -A.P.P.L.E., In Depth — No. 1,* page 16, "The Monitor Input Subroutine".

This concludes our study of where to place a machine code utility. As you can see, there are many good locations, each with something good to offer but none with an iron-clad guarantee of 100% compatability with other programs.

We will continue this study in an early issue and learn how to choose the best storage techniques to build something useful — your own amper-library of Applesoft utilities!

Now Available to A.P.P.L.E. Members
For Only
$11.00

The Elementary Apple

by Bill Sanders

This 10 chapter, easy to read illustrated tutorial delves deep into Apple hardware from **power on.**

- A **must** for anyone getting into Apple computers for the first time.
- Teaches you how to do more with your Apple and get the most out of your programs.
- And Much, Much More.

A.P.P.L.E. Orders
304 Main Ave. S.
Renton, WA 98055

Washington state residents add 7.0% sales tax
Foreign members add $6.00 for shipping/handling
Canadian postage being revised

A Call -A.P.P.L.E. PRODUCT REVIEW:

MX PLUS

Drew Berentes

I have owned an Epson MX-80 for a year and a half now and have greatly appreciated its large number of features. Now that I have added Graftrax, I can print in a variety of densities and type widths, italics, graphics, just about anything I really want to do with a printer. There have been, however, a few flies in the ointment.

Most importantly, these special modes must normally be set from software. This means that I had to go to BASIC whenever I wanted to make a mode change, a genuine pain when working with a locked program, for example. Secondly, when listing programs there was no way to indent the left margin or to skip perforations; this made it hard to put program listings in a binder.

At last, an answer exists. The MXPLUS, at $49.95, is an inexpensive, easy to install hardware modification that makes the Epson printers considerably more versatile and easier to use. The supplier of the MXPLUS is:

Dresselhaus Computer Products
22713 Ventura Blvd. Suite F
Woodland Hills, CA 91364

The MXPLUS consists of a small circuit board, about two inches square. Installation is reasonably simple, comparable to the installation of Graftrax, and requires no permanent modification of the MX-80. The instructions provided are detailed and complete, and contain all needed diagrams. The MX-80 case is opened up by removing four screws from the bottom and lifting up the top half of the shell. PROM 1B is removed from its socket in the MX-80. This is the only difficult part of the procedure, since the pins on the PROM must not be excessively deformed; care and patience are needed. Pin 20 of the PROM is bent out straight from the PROM case, and the PROM is then plugged into a socket on the MXPLUS. Next, the MXPLUS is plugged into the socket that is used to hold PROM 1B. Finally, one wire on the MXPLUS is clipped to the pin that was bent our from PROM 1B, and another wire is clipped onto a pin of one of the ICs on the main Epson circuit board. Following testing, the case is then closed and screwed shut. The procedure requires care, since many delicate wires and parts are exposed, but any reasonably patient person should be able to install the MXPLUS in well under an hour.

The unknowing user would never suspect that your MX-80 had been modified. Until the MXPLUS is turned on, there are absolutely no differences in the operation of the printer. The MXPLUS is turned on by holding down the ON LINE button for more that 1 second. Doing so will cause the MX-80 to beep and the ON LINE light to flash. This done, the user can select from nine MXPLUS operations:

1. Reset
2. Condensed print
3. Double wide print
4. Emphasized print
5. Double-wide print
6. Perf-skip
7. Italics
8. Indentation of left margin
9. 8 lines per inch

A feature is selected by additional presses of the ON LINE button. Each depression produces a beep. Just count the beeps to select a feature; 6 beeps selects perf-skip, for example. Feature selection soon becomes easy and automatic.

When a feature is selected, looking at the ON LINE light will tell if the feature is on or off. If the feature is on, the light blinks "long on — short off." If the feature is off, the light blinks "short on — long off." Pressing the FORM FEED button will toggle the selected feature between states. Finally, depressing the LINE FEED button exits the selection process. Again, if this seems complicated, it soon gets to be automatic and easy.

A feature may be changed at any time, even when the printer is receiving data. Just press the ON LINE button to take the printer off line, and select the features desired. When you go on line again, printing will continue, with the new feature active.

Now, about the features. Most of the Epson print modes are not altered by MXPLUS. The user is just given the opportunity to turn them on from the printer control panel. One significant exception is Double Wide print. Normally, the MX-80 turns off Double Wide print at the end of each line. Thus, to print several lines in Double Wide, appropriate control codes must preface each line to be printed. Frankly, I would rather have Double Wide remain selected until I choose to turn it off, and this is made possible by the MXPLUS. When this mode is turned on by the MXPLUS, it stays on until it is deselected.

The reset option restores the default status to all MX-80 functions except top-of-form. The software reset for the MX-80 also resets top-of-form. Since it is rarely desirable that the top-of-form setting be disturbed, I prefer the approach of the MXPLUS.

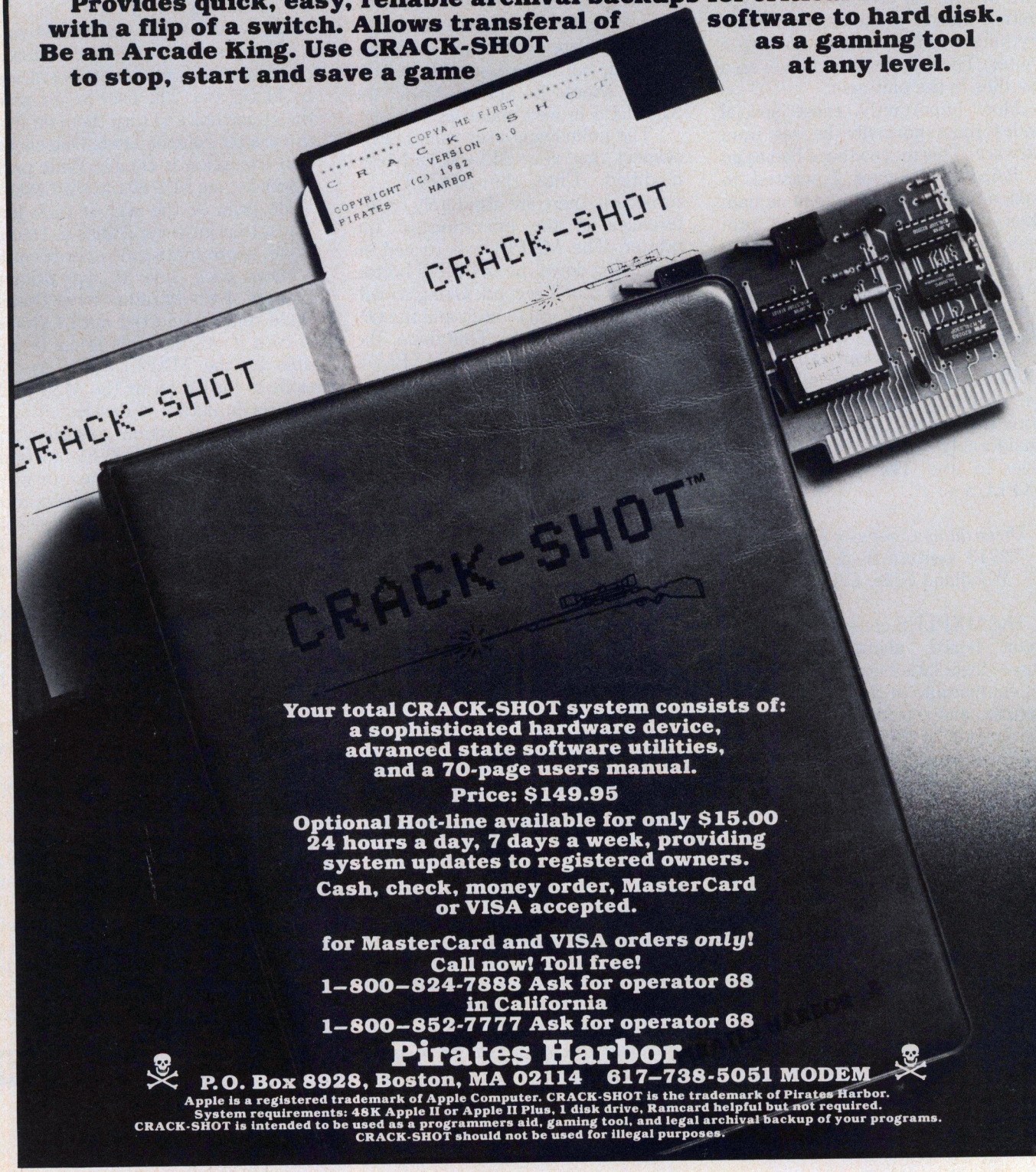
Two of the options on the MXPLUS are not available with a standard Epson. It is possible to select a ten space indent on the left margin. It is also possible to skip perforations at the end of a page; the Perf-Skip option prints six blank lines at the end of every page.

I am greatly pleased with my MXPLUS. For a reasonable cost, it has made my Epson considerably more friendly and useful, and I highly recommend its purchase. One thing the literature does not make clear is whether different versions are needed for different Epson printers. As a precaution, I suggest that you include in your order your printer model and whether or not you have Graftrax, just in case this makes a difference.

CRACK-SHOT™

Internationally acclaimed
The ultimate in copy de-protection

CRACK-SHOT is a total system comprised of hardware and software modules. CRACK-SHOT is easy to use, designed for users and programmers. Eliminates "owners paranoia" about disk failure. Provides quick, easy, reliable archival backups for critical software with a flip of a switch. Allows transferal of software to hard disk. Be an Arcade King. Use CRACK-SHOT as a gaming tool to stop, start and save a game at any level.



Your total CRACK-SHOT system consists of:
a sophisticated hardware device,
advanced state software utilities,
and a 70-page users manual.

Price: $149.95

Optional Hot-line available for only $15.00
24 hours a day, 7 days a week, providing system updates to registered owners.

Cash, check, money order, MasterCard or VISA accepted.

for MasterCard and VISA orders *only*!
Call now! Toll free!
1–800–824-7888 Ask for operator 68
in California
1–800–852-7777 Ask for operator 68

Pirates Harbor

P.O. Box 8928, Boston, MA 02114 617–738-5051 MODEM

Apple is a registered trademark of Apple Computer. CRACK-SHOT is the trademark of Pirates Harbor.
System requirements: 48K Apple II or Apple II Plus, 1 disk drive, Ramcard helpful but not required.
CRACK-SHOT is intended to be used as a programmers aid, gaming tool, and legal archival backup of your programs.
CRACK-SHOT should not be used for illegal purposes.

THE PENULTIMATE PENULTIMATE INPUT-ANYTHING ROUTINE

Bob Nacon

THE normal INPUT command of Applesoft has a very nagging problem of not accepting either commas "," or colons ":"within string input requests. This causes considerable nashing of the teeth amongst data base and mailing list programmers. Basic solutions to the problem involve the GET statement and string concatenation. This method is very slow and uses large amounts of string storage space.

Many articles have been written in *Call -A.P.P.L.E.* and other Apple related magazines proposing different machine language solutions. As time went on better and better solutions were developed. Probably the *best* solution to date (in my opinion) is the Penultimate Input Nearly Anything Routine by Peter Meyer which was published in *Call -A.P.P.L.E. in Depth 1: All About Applesoft* in 1981. This is probably the best Applesoft reference available outside of the Applesoft manual itself. My copy is dogeared from the constant usage made of it.

What Peter Meyer did was produce a machine language subroutine which was fully relocatable, could be called in a number of different ways, and in which any string variable name, including array variables, could be passed to the subroutine for input. There was no need for PEEKing and POKEing address information and he eliminated the need for executing an N$=MID$(X$,1) afterwards to actually capture the string input. The subroutine could be called (if located at $300) by CALL 768:A$. This was really the *penultimate* input routine.

I immediately added it to my checking account program via my own ampersand subroutine program (called AMPER-MAGIC) which I had developed for myself. It significantly improved the speed of entering files from the disk. I began to modify that subroutine to do many of the special functions I needed for my checking account program. I added the capability to prompt for the input by using string literals or string variables, and I can strip off padding characters that I use to fill out fields in a random access file. I can also limit the number of characters entered and ignore all others *before* string storage is obtained for the string. Also, I can enter lower case via the shift key or ESC key. That subroutine grew to become a very powerful and useful input routine.

String Storage

As development continued on my program, I found that I was being eaten alive by the trusty old Applesoft garbage man. This is the routine that is automatically called by Applesoft when less than a specified number of bytes of memory are left for string storage. This memory collection program sequences through the string variables and string array variables looking for all active strings. To back up a moment, Applesoft always uses new string storage space when a string variable is defined. Every time that you do a A$=B$, new string storage is taken equal to the length of B$, the contents of B$ is moved to the new storage location, and the pointers associated with A$ are adjusted to point to the new storage area. Any string assigned to A$ prior to the reassignment is just left in memory as garbage is left in a dump. Applesoft makes no attempt to reclaim this memory at the time of reassignment. At some point, when there is only a little string memory left, Applesoft calls in the garbage collection routine which moves all of the active strings up to the top of memory and reclaims all that unassigned string storage area. Your program may seem to hang for what seems like an eternity while this collection is going on. I was determined to find a solution to the problem.

Every article I have read and everybody I talked to indicated that there was nothing I could do about this string storage problem in Applesoft except to use special garbage collection routines which would speed up the process. I was not really satisfied with this kind of solution, so I kept on reading and experimenting, trying to find reasonable ways of reclaiming string storage memory as new strings were assigned. I wasn't having much success.

Don't Reclaim – Reuse Memory

Then one day it hit me. Why try to reclaim it? Why not *reuse* it? Why not reuse the memory space that stored the old string to store the new string *if* there was room. All I would have to do is compare the length of the old string with the length of the new string and decide whether I could reuse string memory. Then all I would have to do is move the new string into the old string's memory space and adjust the string's length pointer. As it turned out, Peter Meyer's subroutine had all the information needed to make a decision about reusing memory and it would only add 33 more bytes to do it. It worked so well that I wanted to share it with you.

The modification presented here to the Penultimate Input Anything Subroutine compares the length of the new string in the input buffer with the length of the old string and if the length of the new string is *less than* or *equal to* the length of the old string, the new string is moved into the old string's memory locations and the string variable's length pointer is adjusted appropriately. Since we are reusing the same memory space, the variable's address pointers do not have to be changed. If the new string is longer than the old string then the normal string assignment procedure is used. A side benefit of this change is that it takes less execution time to reuse string memory space than it does to get new space and move the string to it.

The addition is clearly identified in the accompanying listing and is well commented. In keeping with the intent of the original subroutine, this addition is fully relocatable so that it can be stored and executed at any address without changes. It can simply be loaded into programs like AMPER-MAGIC and appended to the end of your Applesoft program and CALLed by &INPUT,A$.

FLIGHT FORM

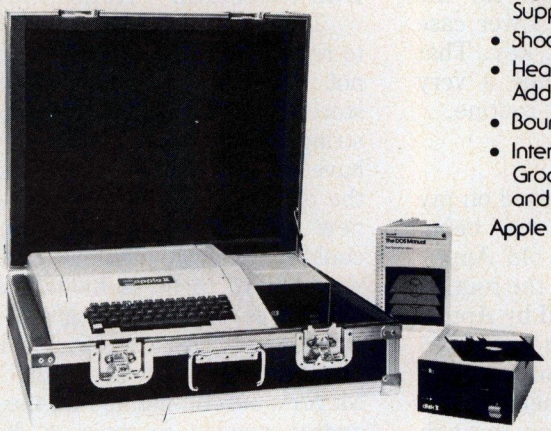
BY CASES, INC.

DESIGNED TO PROTECT YOUR COMPUTER

Features —

- Rigid Shell Made of Plywood Supported High Impact ABS.
- Shock Resistant Foam Lining.
- Heavy Duty Hardware Add $22.50 (U.S. dollars) for lock.
- Bound Metal Edges.
- Interlocking Tongue and Groove Extrusion, Mating Lid and Bottom.

Apple II Plus 2 Drives $175.00



OTHER MODELS AVAILABLE

Overseas Inquiries Welcome

CLUB AND DEALER DISCOUNTS AVAILABLE

CASES, INC. P.O. Box 33820 Seattle, WA 98133-0820 (206) 365-5210

VISA
MASTERCARD

PENULTIMATE

Conclusions and Warnings

The reuse of string memory is a very powerful addition to a very good subroutine. It provides for the most efficient use of string storage space possible. This routine does not eliminate garbage collection, it only extends the time between collection periods. I have used the routine to sequentially input 100 records of a random access file to be outputted to a printer, in which all fields were the same size, and never used more than the initial amount of memory needed for the first record.

There is one warning that must be passed on. *Do not* use it to enter a new value for a string variable which was defined from within a BASIC statement. When you define a string variable from within your Applesoft program like:

```
100  A$ = "TESTING"
```

Applesoft sets up a normal string pointer for A$, but does not use normal string storage areas for the string. Instead it sets up the pointers of the string descriptors back to the word "TESTING" within your program. Now if you use this input routine to enter a new A$ and you enter "1234" as the input, the routine will store "1234" at the old string storage space, which is inside your BASIC program. If you then listed your program you would find

```
100  A$ = "1234ING"
```

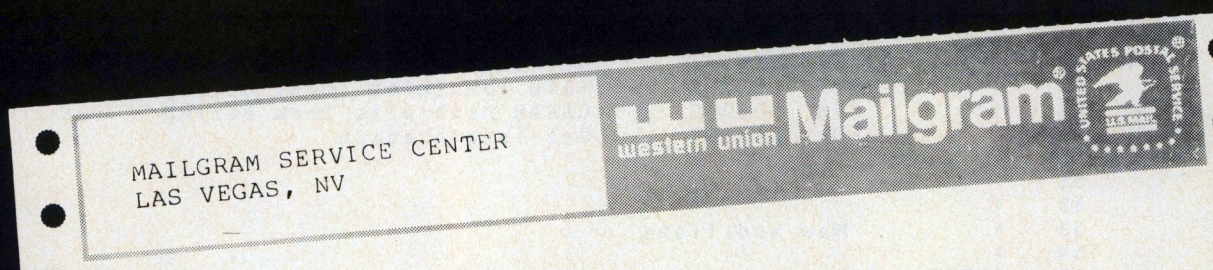
even though the actual value of A$ is just "1234". Remember, we only changed the length byte of the old string's descriptor. This certainly does not cause any trouble until you want to save your program after running it. It's not the same as the original anymore. This is both a problem and a nice feature, as long as you know it is happening and plan for it. Just to be safe though, you should *not* use this routine to enter a new value for any string variable which is defined within your BASIC program.

HAVE FUN!!

```
:ASM

                *********************************
2               *                               *
3               *  An improved version of the   *
4               *                               *
5               *  Penultimate Input Nearly     *
6               *    Anything Subroutine        *
7               *                               *
8               *         by Bob Nacon          *
9               *    All rights reserved        *
10              *                               *
11              * Call -A.P.P.L.E. : Jan 1983   *
12              *                               *
                *********************************
14              *
15              *
16              * Set up JMP $0300 at $3F5
17              * Use & (SVAR$) to CALL it
18              * &A$
19              *
20              * OR
21              *
22              * Setup Amper-Magic for INPUT sub
23              * &INPUT,A$
24              *
25              *
26              * Internal Applesoft CALLs
27              *
28    GDBUFS    EQU   $D539
29    PTRGET    EQU   $DFE3
30    GETSPA    EQU   $E452
31    MOVSTR    EQU   $E5E2
32    NXTCHAR   EQU   $FD75
33              *
34              *Zero page CALLs
35              *
```

```
                        36   FRETOP   EQU  $6F          ;bottom of STRING STORAGE
                        37   VARPNT   EQU  $83          ;used by PTRGET
                        38   LENINS   EQU  $FD          ;length of Input String
                        39   *
                        40   *
                        41            ORG  $300
                        42   *
                        43   *
0300: 20 E3 DF          44            JSR  PTRGET       ;Get address of Descriptor
0303: A2 00             45            LDX  #$00
0305: 20 75 FD          46            JSR  NXTCHAR      ;Get Input String into buffer
0308: 86 FD             47            STX  LENINS       ;SAVE length of Input String
030A: 20 39 D5          48            JSR  GDBUFS       ;CLEAR high bits from string
030D: A5 FD             49            LDA  LENINS       ;Get length again
                        50
                        51
                        52   *
                        53   *          New additions
                        54   *
                        55   INBUF    EQU  $200         ;INPUT buffer location
                        56   STRNG1   EQU  $AB          ;TEMP storage location
                        57   *
                        58   *
                        59   * VARPNT has adr of string descriptor in it
                        60   *
                        61   *
030F: A0 00             62            LDY  #$00         ;Set to look at length byte
0311: D1 83             63            CMP  (VARPNT),Y   ;CMP length of new to old
0313: 90 1D             64            BCC  REUSE        ;reuse memory if new
0315: F0 1B             65            BEQ  REUSE        ;is (=old length
                        66   *
                        67
                        68   *
0317: 20 52 E4          69            JSR  GETSPA       ;Find space in memory
031A: A2 00             70            LDX  #$00
031C: A0 02             71            LDY  #$02         ;sets PTRS to input buffer
031E: 20 E2 E5          72            JSR  MOVSTR       ;moves string to memory
0321: A0 00             73            LDY  #$00         ;Set string desc to new adr
0323: A5 FD             74            LDA  LENINS       ;Get length again
0325: 91 83             75            STA  (VARPNT),Y   ;Set new length
0327: C8                76            INY
0328: A5 6F             77            LDA  FRETOP
032A: 91 83             78            STA  (VARPNT),Y   ;Set low byte of ADR
032C: C8                79            INY
032D: A5 70             80            LDA  FRETOP+1
032F: 91 83             81            STA  (VARPNT),Y   ;Set hi byte
0331: 60                82            RTS
                        83
                        84
                        85   *
                        86   *          New additions
                        87   *
0332: 91 83             88   REUSE    STA  (VARPNT),Y   ;Set new length of old string
0334: C8                89            INY
0335: B1 83             90            LDA  (VARPNT),Y   ;Get low byte of string ADR
0337: 85 AB             91            STA  STRNG1       ;TEMP storage
0339: C8                92            INY
033A: B1 83             93            LDA  (VARPNT),Y   ;Get hi byte of ADR
033C: 85 AC             94            STA  STRNG1+1     ;SAVE hi byte
                        95   *
                        96   * Now move new string to old location
                        97   *
033E: A0 FF             98            LDY  #$FF         ;Set up to move
0340: C8                99   MOVAGN   INY               ;increment for next character
0341: B9 00 02         100            LDA  INBUF,Y      ;Get new char
0344: F0 04            101            BEQ  MOVDON       ;move until end of string
0346: 91 AB            102            STA  (STRNG1),Y   ;Store in old memory location
0348: D0 F6            103            BNE  MOVAGN       ;always
034A: 60               104   MOVDON   RTS               ;Return


--End assembly--

75 bytes

Errors: 0
```

Stop - Don't Buy.



TO: FELLOW APPLE USERS

STOP! DON'T BUY A LETTER QUALITY PRINTER...
UNTIL YOU SEE THE FANTASTIC NEW TRANSTAR 130!
IT'S THE PERFECT COMPANION FOR THE
APPLE: THE FIRST LOW COST DAISY WHEEL
PRINTER WITH

- BIDIRECTIONAL PRINTING
- PROPORTIONAL SPACING
- BOLDFACE, SUPERSCRIPTS, SUBSCRIPTS
- A NEW AUTOLOAD FEATURE
- AND NO SOFTWARE PROBLEMS;
  IT USES EXISTING DIABLO™ ROUTINES

BUT BEST OF ALL IS THE PRICE TAG: ONLY 895!

ISN'T THIS WHAT WE'VE BEEN WAITING FOR?

Transtar

P.O. Box C-96975 Bellevue, Washington 98009

Diablo is a registered trademark of Xerox Corporation.

All Logical Paths Lead to

PDS DATADRIVE™ FOR YOUR APPLE II*

great price

15% greater storage

100% **Apple II*** compatible

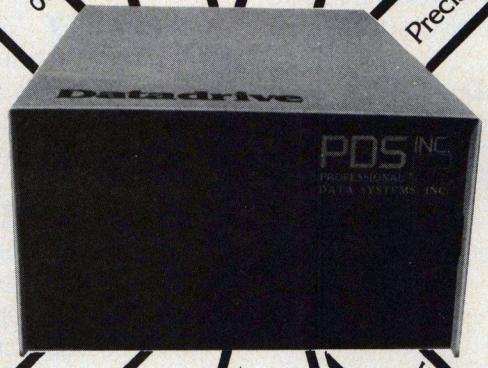
300% track-to-track speed

Outstanding value

Enhancer diskette and 40-track mode

Half-tracking, DOS PASCAL, CPM

Precision lead screen



PDS Universal, Inc.

2630 Walnut Ave., Suite G
Tustin, CA 92680

(714) 730-6772

Telex 18-3511
answer back CSMA

Disconnect Interface Board

When Datadrive is not driving it uses no power

easy interconnect to other computers

Highest standards 8500 hours MTBF

Shugart** mechanism and read/write electronics

conserve power

super-silent operation long life

state-of-the-arts electro-mechanics

*Apple II is a trademark of Apple Computers Inc.
**Shugart is a trademark of Shugart Corporation

ACCESS

SIMPLY THE BEST DATA BASE MANAGER AVAILABLE FOR THE APPLE

- WHY YOU SHOULD BUY ACCESS -

ULTRA-FAST OPERATION
3 Second retrieval on key field. Search for anything, anywhere. ACCESS is 10 to 15 times faster than DB Master and Visifile.

TRUE UPPER AND LOWER CASE
ACCESS supports all 40-column upper and lower case hardware modifications and the Videx Videoterm 80-column card. A pseudo upper/lower case version is also supplied.

LINK TO OUTSIDE WORLD
ACCESS can produce standard DOS 3.3 text files in any format.

SCREEN EDITOR
No need to re-type everything if you make a mistake.

GOOD MANUAL
Tutorial section (tutorial disk supplied), quick reference card, extensive index.

UPGRADES AND ENHANCEMENTS
We listen to feedback from users and are constantly enhancing ACCESS. Free upgrade service to registered users.

CAPACITY
Up to 1521 characters per record
Up to 39 fields per record
Up to 39 characters per field
(79 for Videx version)

SYSTEM REQUIREMENTS
Apple II Plus 48K
1 or 2 disk drives (2 recommended)
DOS 3.3 Disk Operating System
ACCESS supports most makes of printer

ACCESS is available at a special introductory price of **$200.00** including postage and packing from:

Spider Software

98, AVONDALE ROAD, SOUTH CROYDON, SURREY, ENGLAND

Telephone: 01-680 0267
Telex: 943763 Ref: Spider

Your credit card account will be debited for the sterling equivalent at the time of dispatch.

DEALER/DISTRIBUTOR ENQUIRIES WELCOME

THREE DEE DEMOS

David Sun

HAVING studied mathematics and physics at school, it is only natural, I suppose, that one of the first applications of interest to me dealt with three dimensional graphics However, I discovered upon speaking to a few of my fellow "Computors", that although they shared an appreciation for the "pretty pictures," they displayed absolutely no desire to learn the matrix algebra required for 3-D coordinate transformation. So I set out to write a hidden-line, three dimensional image creating program in Applesoft that was both simple to use and easily understood. The result can be found in Listing 1.

```
]LIST

10  HOME : HGR2
20  FOR Z =  - 120 TO
      120 STEP 3
30 XLIMIT =  SQR (14400
      - Z * Z)
40  FOR X =  - XLIMIT
      TO XLIMIT
50 XO =  SQR (X * X
      + Z * Z) * (3.141
      / 100)
60 Y = ( SIN (XO) +
      .75 *  SIN (2
      * XO) + 1.5 *
       SIN (3 * XO))
      * 30
70  GOSUB 100
80  NEXT X: NEXT Z
90  END
100 X1 = 130 + (X +
      Z / 10):Y1 = 100
      - (Y - Z / 10):
      HCOLOR= 3: HPLOT
      X1,Y1: HCOLOR=
      0: HPLOT X1,Y1
      + 1 TO X1,191:
      RETURN
```

The function used as the driver in this example is of the form:

$$Y = SIN(X) + SIN^2(X)$$

The additional numeric values in the equation, (found in line 60), are just modifications and scaling factors which will be discussed later. They are used to proportion the image and to keep it within the limits of the Apple screen. The result of RUNing the program and waiting patiently for about 45 minutes is shown in **Figure 1.** The image is formed using two algorithms, one to give the three dimensional effect and one to produce hidden lines.

How the Program Works

The three dimensional effect is obtained by plotting successive two dimensional images along an imaginary Z axis which propagates directly out of the screen. Line 20 starts the succession at a selected Z-minimum and loops through Z-maximum. Line 30 then uses the Pythagorean theorem to select the X-limit for each Z. This limit increases and decreases according to equal increments on the circumference of a circle. Please refer to **Figure 2** for details.

Line 40 starts the X loop for the limits calculated in line 30. Line 50 takes the radius of the circle at each increment of X and multiplies it by PI(3.141) to create an argument for the function which is contained in line 60. The subroutine at line 100 then plots the calculated points.

By plotting through the limits of X and Z we simulate a three dimensional object which, if you want to look at it another way, is a two dimensional object plotted at Z=0, then rotated 360 degrees around the vertical or "Y" axis.

The second algorithm used is the classic hidden-line algorithm, and is contained in line 120. Here is how it works:

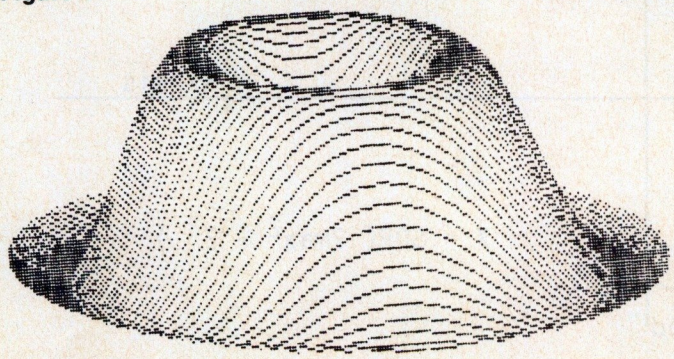
Modifications and Variations

It is intended that the reader experiment with the program, but before making any changes, it is suggested that the following line be added:

```
105 IF X1 < 0 OR X1 > 279
    OR Y1 < 0 OR Y1 > 191
    THEN 130
```

This will cause any points which lie off the screen to be "skipped" thus avoiding program breaks, as the maximum and minimum of many functions are not readily observable.

Figure 1



TRANSFER ///
Version 1.1

A valuable Apple / / / utility

Moves sequential text files either way between an Apple][disk (DOS 3.3) and your Apple ///:

VisiCalc data files
Word-processor files
BASIC programs
as Text files
Laboratory-data files

All actions required are performed easily and automatically. You simply select options from menus.
Runs with 1 or 2 disk drives.

Apple ///
diskette and manual

$60.00

Mass. residents add $3.00 5% sales tax
Overseas orders add $5.00 shipping

Mind Systems Corporation
P.O. Box 506
Northampton, MA 01061
(413)586-6463

Apple][and Apple / / / are Trademarks of Apple Computer, Inc.
VisiCalc is a Trademark of Personal Software, Inc.

THREE DEE DEMOS

The following is a list of the variables and constants used, how they may be altered, and what affect they have on the program:

Z – Ranges from negative to positive Zmax, which is selected by user. Z is used to form X so it should not exceed 120. If it does, the X coordinate may fall off screen at maximum or minimum. Used in line 20 to start loop, line 30 to calculate XLIMIT, line 50 to create argument for function.

14400 – This constant is the square of the maximum value of Z. It is used to calculate XLIMIT. (Figure 2) Should be adjusted according to changes in maximum Z. Found in line 30.

X – Calculated by program. Ranges from –XLIMIT to +XLIMIT. Varies in an inverse manner with Z. Should not be altered by user. Found in line 40.

100 – This is a scaling factor which affects the argument of the function. A lower value creates a larger argument which will make the function more dynamic. Found in line 50 in the form 1/100.

Y – Calculated by program. Value is the main variable upon which the program operates. It can be altered by changing function. Found in line 60.

f(X) – Everything to the right of "Y=" in line 60. This one is up to you. In the example (Listing 1), the constants adjust the relative sizes of the combined curves and the overall image size.

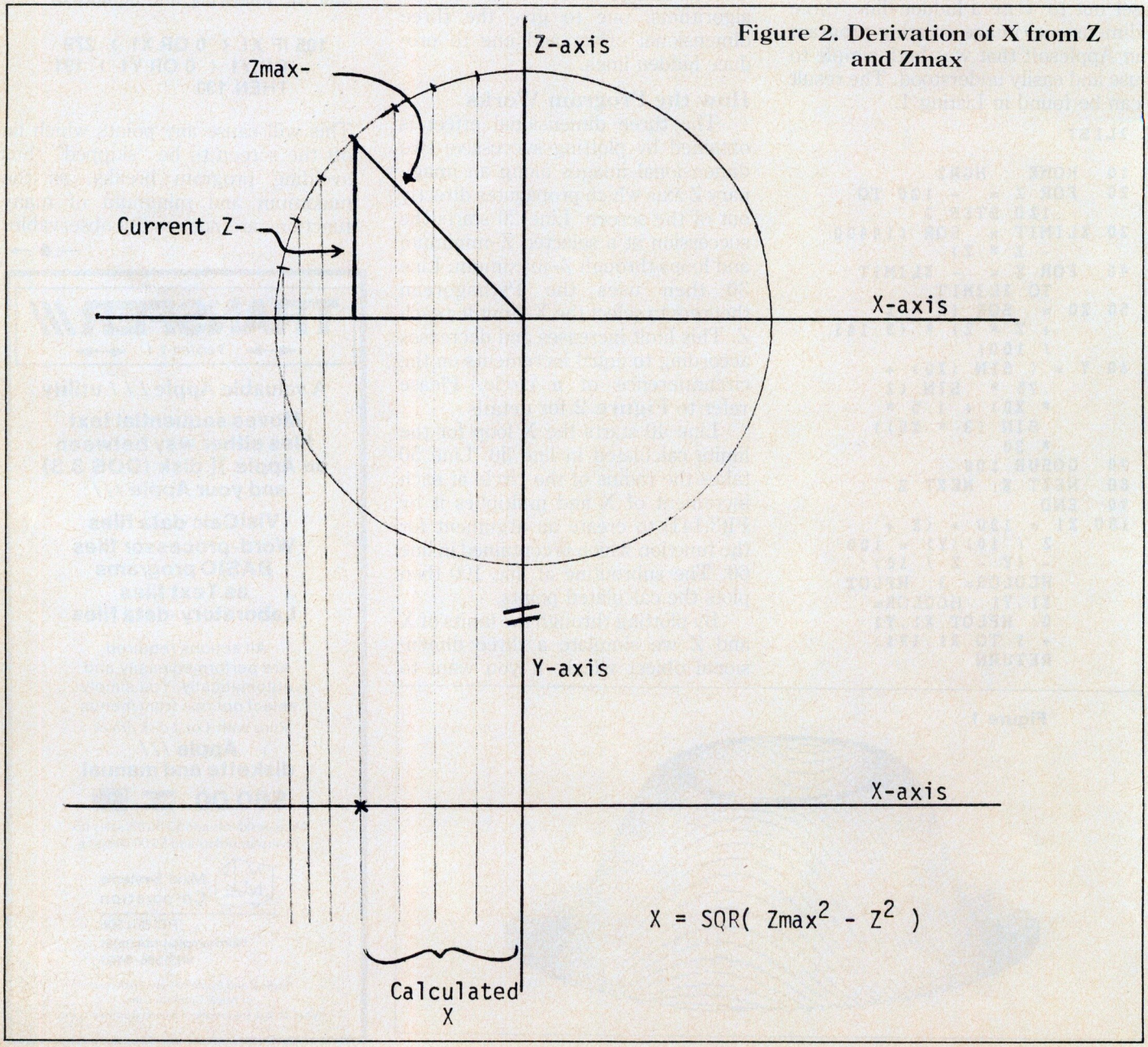


Figure 2. Derivation of X from Z and Zmax

The constants in your equations are up to you.

X1 – The true X–coordinate to be plotted. The factor 130 is used to help center the image on the screen. Remember that we chose to vary X over negative as well as positive values, so it is necessary to move X=0 to the center of the screen. Found in line 100.

Y1 – The true Y–coordinate to be plotted. The factor 100 is used to help center the image on the screen. Since trigonometric functions typically vary in both a positive and negative sense, we again need to move Y=0 to the center. Found in line 100.

Z/4 – This is perhaps the most interesting of all variables and constants. If this variable were not combined with either X1 or Y1, the image would be merely two dimensional. By adding this value, (which increases on each pass of the Z loop), we can effectively "tilt" the image to provide a better three dimensional impression. Adding to X will tilt to the right, subtracting from Y will tilt down. The factor four controls the amount of tilt. Vary this and notice the changes in rotation. Found in line 100.

Finally

I would like to see some reader response detailing their experiments. Perhaps someone would try *color?, inverse images?, machine language?* Maybe a contest to see who can create the best image based on this principal?

The variety of functions is unlimited of course. This type of graphical representation has many applications. Any two dimensional graph can be made into three–D using these algorithms. As a variation you could plot the same curve in succession from "back" to "front" with tilt to create a three dimensional surface over a plane. In a future article I will cover applications more thoroughly.

Here are some samples of variations of the graphs

Figure 3

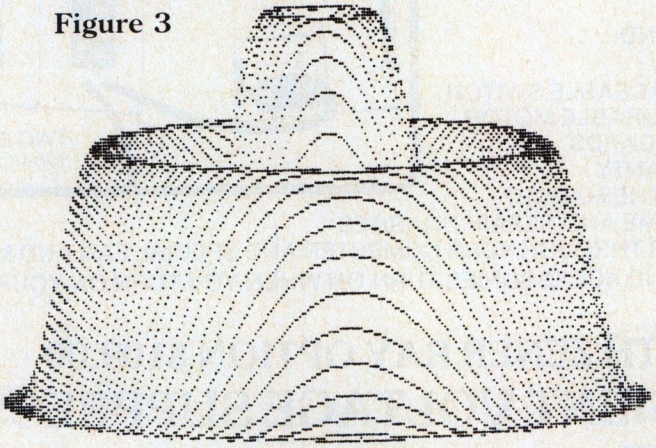
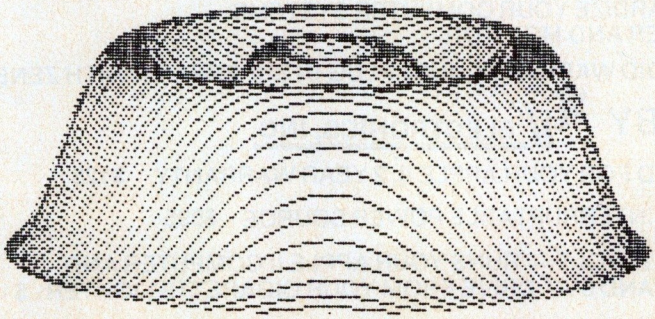


Figure 4



EXPOTEK

2723 W. Windrose • Suite 3
Phoenix, Arizona 85029

1-800-528-8960

GUARANTEED LOW PRICES

APPLE CARDS
16K RAM — $78 Z80 CARD — $245
Videx Card — $227 Smart Term — $279
Micro Soft Prem. PK — $498

ALTOS
5-15D — $2249 5-5D — $4399

ANADEX
9500A — $1290 9501A — $1290

ATARI
400-16K — $289 800-48K — $645
410 — $77 810 — $429
830 — $155 850 — $159

CITOH
F-10P — $1370 F-10S — $1375
1550P — $659 Prowriter 1550CD — $709
8510AP — $479 8510ACD — $570

DATASOUTH
DS120 — $595 DS180 — $1175

DIABLO
630RO — $1945 620 — $1099

APPLE-LOOK-A-LIKE
Call

HAZELTINE
1500 — $995 ESPRIT — $498

MICRO SCI A2
$300 — Apple Drive/Card — $375

MODEMS
HAYS — MICROMODEM — $285
HAYS — SMARTMODEM — $219
HAYS — 1200 Baud — $530

MONITORS
Amdek — Color II RGB — $649
Amdek 300 — $140 Color I — $300
BMC Green — $88 USI Amber — $159

NORTHSTAR
Advantage — $2600 Horizon 64QD — $2600

NEC
8023 — $469 7710 — $2220

OKIDATA
M80 — $305 M83A — $649
M82A — $399 w/Tractor & Grap. $479
M84P — $989 M84S — $1090

TELEVIDEO
802 — $2599 802H — $4450
806 — $4950 800A — $1290

TELEVIDEO
910 — $569 925 — $718
920 — $735 950 — $899

TI
810 — $1240 820 — $1795

ZENITH
Z19 — $689 Z89 — $2129

SOFTWARE
All Major Brands — $CALL

DISKETTES/BOXES
Elephant — $20 Scotch — $25 Dysan — $35

All Prices Subject To Change

Customer Service
602-863-0759

RH ELECTRONICS, INC.

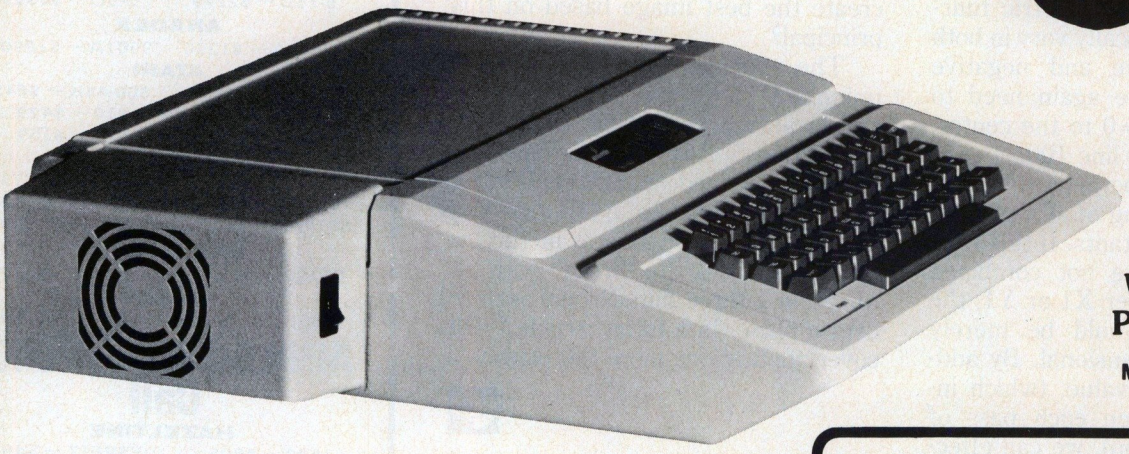
COMPUTER PRODUCTS
DESIGNING • MANUFACTURING
ELECTRONIC ENGINEERING

COPYRIGHT © 1981 — PATENTS PENDING

566 Irelan, Buellton, CA 93427
(805) 688-2047
8:00 TO 5:00 CALIFORNIA TIME

SUPER FAN II™

FOR YOUR APPLE II* COMPUTER

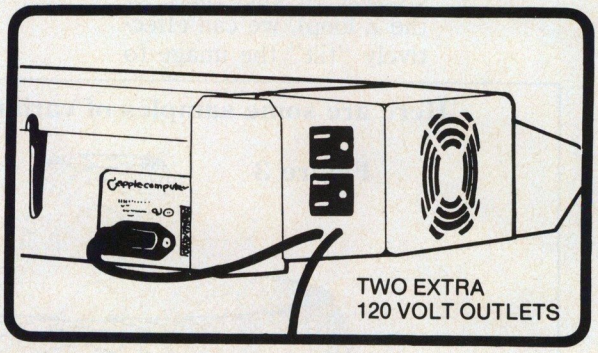


One Year Warranty

$74.95

With Zener Ray™ Protection $109.00

MASTERCARD — VISA



"COOL IT"

- ALSO FITS ON APPLE'S* NEW MONITOR STAND
- RED PILOT LIGHT ON/OFF SYSTEM SWITCH
- CLIPS ON — NO HOLES OR SCREWS • REPLACEABLE SWITCH
- AVAILABLE IN 120V or 240V AND 50/60 HZ • DURABLE MOTOR
- REDUCES HEAT CAUSED BY EXTRA PLUG-IN CARDS
- SOLD WORLD WIDE • UNIQUE 1 YEAR WARRANTY
- TAN OR BLACK COLOR • QUIETEST FAN ON THE MARKET
- INCREASED RELIABILITY — SAVES DOWN TIME AND REPAIR CHARGES
- LOW NOISE DUE TO DRAWING EFFECT OF AIR THROUGH YOUR COMPUTER AND SPECIAL FAN AND MOTOR DESIGN
- TWO EXTRA 120V OUTLETS FOR MONITOR AND ACCESSORIES TURN ON WHEN YOU TURN ON YOUR FAN (NOT AVAILABLE ON 240V MODEL)

SUPER FAN II™ WITH ZENER RAY OPTION $109.00

ZENER RAY™ TRANSIENT VOLTAGE SUPPRESSOR

OUR BUILT IN ADVANCED DESIGN UNIT GIVES
DRAMATIC COST SAVINGS — STOPS ANNOYING DOWN TIME
INSURANCE FROM VOLTAGE SPIKES - GLITCHES

DANGEROUS VOLTAGE SPIKES CAN JEOPARDIZE YOUR COMPUTER SYSTEMS
PROTECT COMPUTER - DISK DRIVE - PRINTER AND MONITOR

NO CUTTING WIRES • WON'T VOID WARRANTY, JUST PLUG IN SUPERFAN II WITH ZENER RAY

OTHER PRODUCTS BY RH ELECTRONICS, INC.

SUPER RAM II™ 16K RAM CARD FOR YOUR APPLE II. 2 YEAR WARRANTY $125

GUARDIAN ANGEL™ AN UNINTERRUPTABLE POWER SOURCE $595

12 VOLT TRANSVERTER 12 VOLT — RUNS YOUR APPLE II COMPUTER AND AND 5¼" DRIVE FROM YOUR CIGARETTE LIGHTER $149

*Registered trademarks of Apple Computer Inc.

DEALER INQUIRIES INVITED

A Call -A.P.P.L.E. TECHNIQUE:

USING THE SPLITTER

Norman L. Kushnick

Background

THIS is written for those Apple owners who, like myself, do not talk machine language. This background section deals with the Apple's memory map and is not essential to using the "Splitter" program, but will help you understand the problem and the solution offered by that program. A "hex dump" of Splitter appears in conjunction with this article.

In issue one of *Call —A.P.P.L.E. in Depth: All About Applesoft,* an article by Charles Kluepfel appeared in which he presented a program called "Splitter." However, for non-technical types like me, it was very difficult to learn how to successfully use this very powerful machine language program.

Assuming you have a 48K Apple][+, when you first turn your computer on, you have available to you all the memory from byte 2049 to 49151. However, once you load DOS by booting a disk, you have lost about 10K at the high end of memory and now have from byte 2049 to 38399 available. This is still over 36K which is a lot of memory for almost any program.

As you type in your program from the keyboard or load it from disk, it starts at location 2049 and builds upward from there. Math variables are placed above the highest program location, and string variables start at byte number 38399 (which is equal to HIMEM–1) and build downward. Again, in most cases, there is no problem.

The problem usually occurs when your program is using one of the high resolution graphics pages. The command HGR uses page 1 of the Hi-Res graphics screen in which you have four lines of text available at the bottom; and HGR2 uses Hi-Res graphics page 2 where the graphics go to the bottom of the screen.

In the Apple, these "pages" are at specific memory locations. Page 1 (HGR) occupies the space from location 8192 to 16383; and HGR2 occupies the space from 16384 to 24575. These memory allocations are fixed within the Apple and there is nothing you can do to move them to some other location in memory.

The Problem

The problem will most often occur when you are writing an Applesoft program that uses the HGR portion of memory to produce graphics. Assuming that you develop your program in stages, testing as you go, here is what may happen.

Remember that your program starts loading at location 2049 and builds upward, with math variables stacked on top of the program itself. (String variables start at 38399 and build downward and seldom cause the problem.)

As long as your program, plus the math variables, will fit between locations 2049 and 8191, there is no problem.

However, as your program grows, it may shove the math variables above location 8191. Now the math variables are getting into the portion of memory reserved for page 1 of Hi-Res graphics. If your program contains an HGR command, funny things will begin to happen – you will begin to get random horizontal line segments appearing on the graphics screen.

However, at this stage, your program will still run, but the random lines are messing up your screen. If your program is completed at this stage, you can avoid the messed up screen without using the Splitter program by inserting the following command as the first statement in your program: **LOMEM : 16384.**

The LOMEM command makes the *math variables* move to location 16384 and above, thereby getting them off of the area reserved for page 1 graphics. However, LOMEM does not move the program itself.

After adding that command, your program should run without difficulty.

Here is what you have at this point: The program is contained between locations 2049 and 8191, the math variables are at location 16384 and above, and the string variables continue to be up in the 38000 range. Since the memory locations between 8192 and 16383 are not used by any of these, everything is working smoothly.

But now let's assume that your program continues to grow. Remember that your program begins at location 2052 and builds upwards. Once the *program itself* extends beyond location 8191, stranger things happen – the program bombs! When you load the program, all of it will LIST. But after you RUN it, statements at the end will have disappeared if you then LIST the program.

The execution of the HGR command within the program destroys the portion of the program which extended above location 8191. Since it is not always practical for your program to load anywhere other than at location 2049, and since there is no way to move the start of the HGR portion of memory above 8192, you now have a Problem with a capital P.

SIXTH FINGER™

The automatic repeat key for your Apple II*

$9.95

Introductory price

»Adapts your Apple; all keys held down will repeat after the perfect delay.

»Great for program editing, word processing & VisiCalc*

»Easy for anyone to install. Plugs onto the Apple's encoder board (all rev. 7 or later and late rev. 6 with encoder board).

(30 day trial)

Special introductory price $9.95
add shipping & handling 1.00
Cal. residents add 6% tax .60
Send check or money order to:

Ivers Specialties
Dept. CA-1
12841 Hawthorne Blvd.
Box 524
Hawthorne, Ca. 90250

Dealer Inquires Invited

*Apple is a registered trademark of Apple Computer Co.
*VisiCalc is a registered trademark of VisiCorp.

USING THE SPLITTER

You have all that memory available from 16384 to 38400 which is now occupied only by your variables, but no apparent way to get to use that large portion of your memory for your program.

Here is where the Splitter program comes to the rescue. And here's what it does:

After following the steps in the next section, here is where your program and variables will be located: As much of the program as will fit still resides in locations 2049 to 8191. The rest of the program is located at 16384 and above, along with the variables. You will have split the program to leave locations 8192 to 16383 unused so that when HGR is executed, there is nothing to interfere with that part of memory. Now your program which previously was limited to about 6K in length (2049 to 8191) can now be much larger since the memory above 16383 is now available.

Entering and Saving the Splitter Program

The article which accompanied the program explained in great depth what the program steps do, but it lacked a clear example of how to use it. Hopefully, what follows will rectify that problem in non-technical language.

First, remove the LOMEM command from your program, if you added one, and save the revised program.

Next, you must enter the program appended to this article and save it on the same disk. Here is how you do that (omit the quotes):

1. Type "CALL -151" and hit return. You will get an asterisk and are now in the monitor.
2. Type "9400:" and then begin typing each pair of numbers and letters shown in the listing, leaving a space between each pair.
3. After you have entered two or three lines, hit return and continue entering the pairs of characters, hitting return every couple of lines, until you get to the end of the program listing.
4. Now type "9400L" and hit return. A screen-full of the program will list. Check it against the listing for errors. Suppose an error is found at location 9408. Enter "9408:" followed by the correct characters and hit return. Now repeat this step. When the first page is correct, type "L" and hit return again and another screen-full will appear. Continue until you see all zeros and have reached the end of the program with all errors corrected.
5. Now save the program on your disk with the following command:

   BSAVE SPLITTER,A37888,L270

   37888 is the same in decimal as 9400 in hex, and 270 is the length of the program in bytes. Enter the above command and hit return. SPLITTER is now saved on your disk.

Using the Splitter Program

Let's assume your program, which uses page 1 of Hi-Res graphics, is called GAME and is on the same disk with the SPLITTER program you just entered and BSAVE'd.

Before you use the Splitter program, you must tell it two things:

1. How big the split is to be, and
2. Where the split is to begin.

Here's how we do that. The area we want to leave blank is from 8192 to 16384 which, if you subtract, consists of 8192 bytes. Divide that number by 256, and get 32 as the answer. We must put this value (32) into memory location 6.

We want the split to begin at location 8192 which, when divided by 256, comes out to 32. (It is only coincidental that both values come to 32 - the HGR page is 8192 bytes long and also starts at location 8192.) We must put this second value (32) into memory location 7.

(If it were HGR2 we were trying to protect, we would put 32 into location 6 and 64 into location 7. (24576-16384)/256 = 32 and 16384/256 = 64).

After we put those values into the proper location, we BLOAD the Splitter program from the disk to memory. Then we LOAD GAME as usual, then execute the Splitter program, and finally RUN will execute GAME.

Here are the commands needed to accomplish this:

```
POKE 6,32 : POKE 7,32
BLOAD SPLITTER
LOAD GAME
CALL 37888
RUN
```

The POKE's put the proper values into locations 6 and 7 as determined above. BLOAD SPLITTER loads the Splitter program into memory. The BSAVE command we used when we saved the program makes this program load beginning at location 37888, the top end of available memory.

LOAD GAME loads your program into memory. At this point, it extends into the HGR page, but since you haven't RUN it yet which would execute the HGR command, no harm is done.

CALL 37888 executes the Splitter program and splits GAME so that none of it is on the HGR page. Finally, RUN executes GAME.

Actually, the first three commands above can be put into a little program which would be executed first. This would make the required POKE's, BLOAD Splitter and LOAD GAME. Then the last two commands are entered from the keyboard.

Here is such a program, which might be called USE GAME:

```
10  REM  USE GAME
20  D$ =  CHR$ (4)
30  HOME : VTAB 3: PRINT
      "WHEN CURSOR FLASHES,
      ENTER FOLLOWING
      DIRECT COMMANDS:"
40  PRINT : PRINT "CALL
      37888"
50  PRINT : PRINT "RUN"
60  POKE 6,32: POKE
      7,32
70  PRINT D$;"BLOAD
      SPLITTER"
80  PRINT D$;"LOAD GAME"
90  END
```

Final Notes

Your GAME program cannot be saved in its split configuration. If you do so, you will simply use more disk space, but if you try to RUN it directly from the disk without using the USE GAME program, the split will not be maintained and the program will bomb.

You can make changes in GAME by LOADING it from disk and, after making the changes, save it on disk

again. However, you cannot execute (RUN) it without executing the USE GAME program first.

If GAME does not yet extend beyond location 8191, then when you type "CALL 37888" the system will hang up. However, no harm has been done; just hit RESET, enter RUN and hit return.

I hope these notes will help other Apple owners and will allow them to avoid several days of total frustration, which I experienced, trying to use the Splitter program successfully.

Listing 1
Splitter Hex Dump

```
*9400.950C
9400- 38 A5 B0 85 41 65 06 85
9408- 43 A5 AF 85 40 85 42 A0
9410- 06 A9 00 91 42 68 D0 F9
9418- B1 40 C6 40 91 42 C6 42
9420- A5 42 C9 FF D0 F2 A5 41
9428- C6 43 C6 41 C5 07 10 E8
9430- A5 07 85 08 C6 08 38 A5
9438- B0 65 06 85 B0 85 6A 85
9440- 6C A5 67 85 40 A5 68 85
9448- 41 A0 01 B1 40 C5 08 F0
9450- 0B AA 88 B1 40 85 40 86
9458- 41 C8 D0 EF 88 B1 40 F0
9460- 02 D0 05 C8 B1 40 D0 E9
9468- C8 C8 C8 C8 B1 40 D0 FB
9470- C8 C8 C8 C8 C8 A9 AB 91
9478- 40 A5 40 85 42 A5 41 85
9480- 43 C8 84 09 A0 01 B1 40
9488- 38 65 06 91 40 88 AA B1
9490- 40 85 40 86 41 C8 A6 09
9498- D0 11 B1 40 D0 EA 4C 00
94A0- 00 10 27 E8 03 64 00 0A
94A8- 00 01 00 C8 B1 40 85 19
94B0- C8 B1 40 85 1A A9 A1 85
94B8- 1B A9 94 85 1C A0 01 A2
94C0- 01 A9 2F 85 1D E6 1D A5
94C8- 19 88 38 F1 1B 85 19 C8
94D0- A5 1A 30 02 A2 00 F1 1B
94D8- 85 1A 10 E9 8A D0 E6 A5
94E0- 19 88 18 71 1B 85 19 C8
94E8- A5 1A 71 1B 85 1A 84 1E
94F0- A5 1D A4 09 91 42 C8 A9
94F8- 00 91 42 E6 09 A4 1E C8
9500- C8 98 C9 0A 30 BB 86 09
9508- A0 01 4C 9A 94
```

A Call —A.P.P.L.E. QUICKIE:

WordStar and the Apple][

Steve Ross

I can offer an enhancement for those of you with Videx, Sup'r'Term, and Smarterm's. The ordinary CP/M installation of Wordstar provide for highlight and lowlight text to make the menus and blocks more distinct. We ordinarily don't see that. Here's how to install the High/low-lights: Run install as you normally would, but when it asks if you're done, answer N. Next modify the following addresses as follows:
IVON 284 = 02, 1B, 29
IVOFF 28B = 02, 1B, 28 (switch inverse off)
You might not like this feature, so install it as something new, or back up your last installation.

A.P.P.L.E. Members Only

INTRODUCTORY OFFER

Ap.Graphics
The Filler

A set of machine language routines that quickly fill Hi-Res areas with colors or patterns.

The Filler offers:

- Picture/Pattern Editor
- Ampersand Vectored
- Extremely Fast
- 21 Colors with Pre-defined Patterns

Member Introductory Price:
$17.00
(reg. $20.00)

A.P.P.L.E. Orders
304 Main Ave. S.
Renton, WA 98055

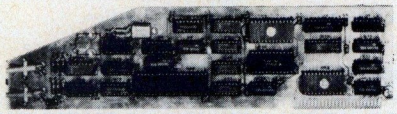
Foreign members add $6.00 for shipping/handling
Washington state residents add 7.0% sales tax
Canadian members, postage being revised

Offer expires March 1, 1983

219.95

FINALLY A RIGHT PRICE FOR THE RIGHT PRODUCT

micromax
presents
VIEWMAX-80



THE RIGHT 80-COLUMN VIDEO CARD FOR YOUR APPLE II*

It is more compatible with existing 80-column application software than other leading cards. And it is loaded with features. Check for yourself.

	RETAIL PRICE	7 X 9 CHAR DOT MATRIX	TRUE DESCENDERS	BUILT-IN SOFT VIDEO SWITCH	BUILT-IN INVERSE VIDEO CHAR.	FULLY PASCAL COMPATIBLE	SHIFT KEY SUPPORT
VIEWMAX-80	219	Yes	Yes	Yes	Yes	Yes	Yes
SMARTERM*	360	No	No	Yes	Yes	Yes	Yes
VIDEOTERM*	345	Yes	Yes	No	No	No	No
SUP'R TERM*	375	No	Yes	No	Yes	No	Yes
OMNIVISION*	295	No	No	No	Yes	No	Yes
VISION 80*	375	Yes	Yes	Yes	Yes	Yes	Yes
FULLVIEW 80*	379	Yes	Yes	Yes	Yes	Yes	Yes
WIZARD 80*	245	Yes	Yes	No	Yes	Yes	No

VIEWMAX-80 is functionally compatible with VIDEX's Videoterm and works easily with 80-column applications like CP/M, Pascal, WORDSTAR, AppleWriter II, Zardex, Easywriter, Letter Perfect, Magic Window, Apple PIE, VISICALC, ASCII EXPRESS, Transend, dBASE II, etc.

Shift Key Mod and Preboot disk for Visicalc and AppleWriter II available as low cost options. See your local Dealer or order direct. Visa/MasterCard accepted. Dealer & OEM inquiries welcome.
BACKED BY A FULL 2-YEAR WARRANTY.
DEALER & OEM INQUIRIES WELCOME.

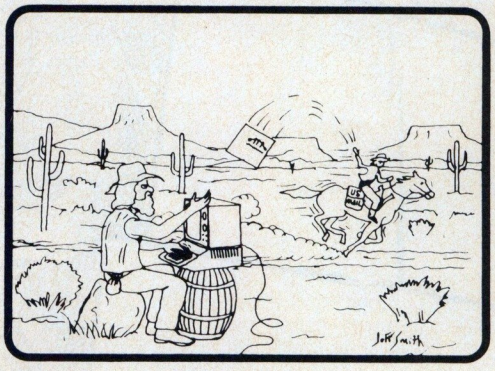
micromax
1130 camino del mar, suite F
del mar, ca 92014
(619) 569-2121

Also, Apple Parallel Printer Card; $79.95
Apple 40-Column Upper/Lowercase Adapter: $29.95
Z-80 CP/M Card: $149.95

"WE MAXIMIZE YOUR MICRO INVESTMENT"

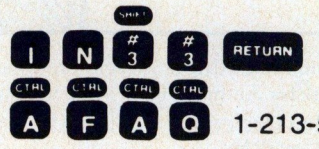
*Trademarks of Apple Inc, Advanced Logic Sys., Videx Corp, M&R Enterprises, Computer Stop, Vista Computers, Bit 3 Computers, Weaper Microsystems.

TELEPHONE SOFTWARE CONNECTION™ INC



No MODEM yet??? We'll find another way to reach you! Send for our free catalog.

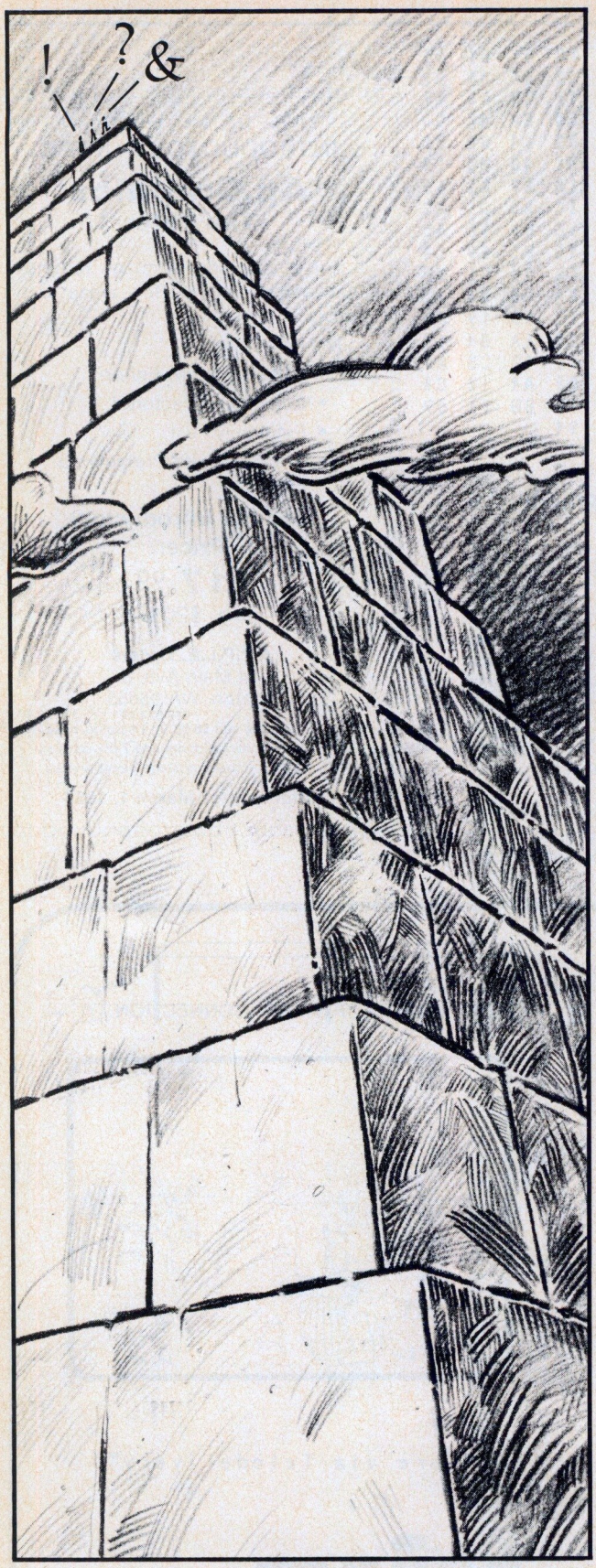
Software via Teledelivery™!



1-213-516-9432

POST OFFICE BOX 6548 - TORRANCE, CALIFORNIA 90504

© 1982 Telephone Software Connection, Inc.



We've moved ahead a byte in time since the Tower of Babel. And at least three things have improved.

Here they are:

Thing Number One: Our Communication's Improved.

With AMPER-MAGIC™ there's no need to know machine language to benefit from its power.

To your Applesoft™ programs you can attach slick, finished machine language routines (referenced by name, not by address, no less) in seconds. No separate BLOADing. And then, with AMPER-MAGIC commands, you can pass variables back and forth directly, just as you would with built-in Applesoft commands.

So, name your routine, perform the append procedure (only once please), and AMPER-MAGIC takes over — generating machine language speed where it counts: in *your* programs. Even your commercial programs. And, if for some reason you want to remove the routine, AMPER-MAGIC can do this too.

Thing Number Two: Our Communication's Improved.

AMPER-MAGIC also gives you access to hundreds of relocatable machine language routines, including those from magazines and other libraries, and - most notably - those from AMPER-MAGIC itself, which has over fifty routines of its own in two volumes.

Some of AMPER-MAGIC's routines are:

Find substring	Move memory	Multiple poke
Speed up Applesoft	Input anything	Swap variables

And, of course, there are others too, for business, educational, recreational and word-process uses.

With AMPER-MAGIC you attach and access. It's really that simple. You can attach an unlimited number of commands to any program and you can move any set of commands as a unit to any other program.

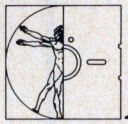
A marvelous simplicity, wouldn't you say?

Thing Number Three: Our Communication's Improved.

Compatible with AMPER-MAGIC you can get the AMPER-MAGIC COMMAND LIBRARY. Already available is Volume 2, which contains twenty-seven machine language routines devoted to information displays and output capabilities.

With Volume 2 you get: a WAIT command, which specifies time in ordinary tenths of seconds either as a variable or as an expression which can be controlled by the program; the most powerful PRINT USING command yet devised; and, of course, twenty-five other commands. Volume 3 will be devoted to Memory Management including a way to store 100,000 numbers in a 48K Apple.

Imagine!
Only $75.00 for the original AMPER-MAGIC
Only $35.00 for AMPER-MAGIC COMMAND LIBRARY, Volume 2
See your dealer or contact us.



Anthro-Digital

"The Company That Interfaces People and Computers"
103 Bartlett Avenue, Pittsfield, MA. 01201 413-448-8278

For those of you who would like AMPER-Magic's routines in your own programs, just ask for Anthro-Digital's free-use policy.

AMPER-MAGIC and AMPER-MAGIC COMMAND LIBRARY, Volume 2 are trademarks of Anthro-Digital, Inc.
Applesoft is a trademark of Apple Computers, Inc.

Also available from Anthro-Digital: Versacalc (several versions), Visi-Caids, The Executive Secretary, The Personal Secretary, and Executive Speller, Hebrew II and Hebrew II Plus, S-C Macro Assembler, Quick Trace, Amper-Magic, The Rental Manager, F.A.R.M., The Performance Manager, Omniscan VideoDisc interface, and Flipper. DEALERS INVITED

DIRECT TO PAGE TWO WITH APPLESOFT

Bob Halfman

OVER the past year, as part of the learning process on my son's Apple][Plus, I have been intrigued by the various attempts to put text directly on the secondary display page, page 2, (See Apple Ref. Manual p.12) as described in the *Call -A.P.P.L.E.* June '80 (p.7). This topic appeared in other earlier issues and most recently in *Apple][Monitors Peeled* (p.47).

In putting text on page 2, the basic difficulty lies in the inflexibility with which the monitor routine BASCALC at $FBC1 (Ref. Manual p.146) insists on page 1 by calculating memory locations only in the range from $400 to $7FF as corresponding to the desired screen line positions. Since the BASCALC routine is only called when a new screen line is desired, such as with a carriage return or after reaching the edge of the window, previous attempts to write directly on page 2 ($800-$BFF) have interfered only when the line change is called for. Alternatively the entries are made on page 1, then moved in bulk to page 2.

After reading "Wizardry with the System Monitor" in *Creative Computing* (Dec 81 p.194), I realized that almost all text going to the screen from the keyboard, Applesoft or the monitor goes through the COUT ($FDED) routine to COUT1 ($FDF0) by way of the Character Output Switch represented by the indirect jump with zero-page address CSWL,H ($36,$37.)

This is one of the locations where data and commands can be diverted to peripherals such as disk and printers. By intercepting our text there every character going through CSW to the screen can be checked for a page 2 address and modified when necessary. List 1 shows the fundamental assembly language program for the interception process which looks at the high byte BASH $29 of the screen address and, if less than $08, adds $04.

This takes a page 1 address between $400 and $7FF and converts it to the corresponding page 2 address between $800 and $BFF. Since the contents of the A, X and Y registers must be preserved so that complex output routines such as PRA1 ($FD92) are possible, the contents of A are pushed on the stack at the beginning of List 1 and pulled back to A before jumping to COUT1 ($FDF0) and thus to the screen; X and Y are not disturbed at all.

List 1

```
*300L

0300-   48          PHA
0301-   A5 29       LDA   $29
0303-   C9 08       CMP   #$08
0305-   B0 04       BCS   $030B
0307-   69 04       ADC   #$04
0309-   85 29       STA   $29
030B-   68          PLA
030C-   4C F0 FD    JMP   $FDF0
```

```
1   POKE 54,240: POKE
      55,253: POKE 49236,0:
      END
2   POKE 54,0: POKE 55,3:
      POKE 49237,0:
      END
```

Fortunately, this idea proves quite compatible with DOS use of the same type of interception. At the time of this development we had not yet invested in a disk drive and had only very recently acquired a printer. You may notice that List 1 is nothing more than output of the Apple Mini-assembler. We had no fancy assembler so we made a relocatable version of the miniassembler program listed in *Creative Computing* (Oct. 81 p.224). I then added a stepper lifted from the Monitor ROM Listing at the back of the Reference Manual (p.155) to help in debugging code and combined a titler with the mini-assembler.

Getting back to the use of the List 1 routine, it is convenient to enter in monitor mode the machine code starting at $300 and also change the CSW values at $36 and $37 from $F0 and $FD for COUT1 to $00 and $03 for the low and high bytes of $300. One further step is needed to display page 2. You must activate the screen display soft switch (Ref. Manual p.12) for page 2 by typing $C055 in monitor mode with a return.

You can use a very simple Applesoft program to experiment with this page switching, but only after you prevent Applesoft from beginning its program at the usual $800 right in page 2 memory. To move Applesoft to $C00 above page 2, the pointers for the program beginning and end must be changed by typing in monitor mode 68:0C and B0:0C (See *Call -A.P.P.L.E. in Depth* #1 p.5). Memory locations $67 and $AF should read 01 and 03 or 04 respectively. You must also type C00:00 00 00 for Applesoft programs to be entered successfully from the keyboard starting at $C00. A single zero at $C00 is adequate to ensure proper loading of a program from disk.

You can now type the Applesoft program in List 1 on either page 1 or 2. The command RUN 1 will put (in decimal entry) $F0 and $FD in CSWL,H ($36,$37) and trigger page 1 display by POKEing $C054. RUN 2 will enter $00 and $03 in the CSW address to send the screen input through the List 1 program and will trigger the page 2 display with $C055.

As you can see with some experimentation, entry of text to page 2 from Applesoft, monitor or keyboard works now, but you may have a few difficulties. Clearing the screen is a problem, as HOME seems to work on page 1, even when commanded from page 2. Also, when you get to the bottom of page 2 there is no scrolling action there; it too occurs on page 1. A VTAB 0 command will move the cursor to the top of page 2 but otherwise Applesoft is quite helpless.

The monitor can do a bit better. If you enter on page 2 in monitor mode 800:A0 N 801<800.BFEM N 25:0 N E003G and hit return, you will have a clear page 2 with cursor at the top. The 800:A0 puts code for a blank in $800 and the monitor MOVE command 801<800.BFEM spreads it all the way to $BFF (See Ref. Manual p.56). The 25:0 puts 0 in CV ($25) for top line and E003G calls for return to Applesoft from the monitor. Note that N merely separates commands for the monitor much like : does for Applesoft.

DIRECT TO PAGE TWO WITH APPLESOFT

In order to use page 2 effectively for text, the SCROLL and HOME problems must be solved. I therefore looked at the monitor routines at $FC70 and $FC58 and noticed that they call BASCALC ($FBC1) through VTAB ($FC22) or VTABZ ($FC24) (Ref. Manual p.146) repeatedly without going through COUT and thus our CSW intercept. SCROLL simply moves lines up one at a time, character by character, using appropriate memory ranges for each line according to BASCALC. Since we cannot change the monitor ROM value at $FBC6 from $04 to $08, we must have our own scroll and HOME routines which can call BASCALC ($FBC1) and then modify its output for page 2. This we do in List 2.

Lines $C00 to $C15 of List 2 incorporate our List 1 intercept into a P2VTAB or P2VTABZ for page 2 so the intercept routine can be used both as part of a BASCALC called from P2SCROLL or P2HOME and as a subroutine between INPUT at $C16 and OUTPUT at $C2D (with the A,X and Y registers preserved). Thus a character headed for page 2 screen comes from COUT to INPUT and is checked for a page 2 address. Then the Y register is saved in YSAV ($35), the input line CV ($25) is compared with the bottom of the scroll window $23 and, if no scroll is needed, the character is sent to COUT1 and thus to the screen after Y is restored. (For an excellent listing of zero page fields see *Monitor Peeled* p.3-13.)

A scroll will be needed if, on the bottom line, either we find we have reached the end of the line by comparing $24 with $21 or if a carriage return $8D is entered. In both of these cases we want to go to P2SCROLL at $C3D, but in the first case we must first print at $C3B the end-of-line character (after ensuring that it is printable - that is, not between $80 and $A0). In the second case in moving to P2SCROLL after a carriage return, we often want to have the [Ctrl-S] command available to stop scrolling when looking at a long program list or a large memory dump. Thus before scrolling, we insert the subroutine from lines $C9E to $CB0 which checks the keyboard for the Ctrl-S ($93) and, if found, sits in a loop until nudged from the keyboard.

This is basically the VIDWAIT ($FB78) routine from the monitor. The P2SCROLL and P2HOME routines are largely transcribed from the monitor as well, except that lines $C94-$C9A, while preserving the A register, put a 2 in the lower right corner of the screen to identify page 2 operations.

Although page 2 text now enters quite well in a controllable text window with left and right arrows, [Ctrl-X], [Ctrl-S], scroll and HOME in operation, the Escape-IJKM cursor moves used for editing bypass the CSW intercept and disappear to page 1 when we attempt to change lines. If this edit feature is wanted on page 2, the only way I found to produce it is to note that it pertains only to keyboard input, can be intercepted at the Keyboard Input Switch jump locations KSWL,H $38,$39 (Ref. Manual p.83) and sent to P2EDIT at $CB3 for possible modification.

If we check at the time of each intercepted keyboard input for the screen address in BASH $29 for the character previously handled, we should always get page 2 unless the previous input was in Escape mode which, through the monitor at $FC2C, put the blinking cursor on page 1. Having identified this situation, we can unblink page 1 with line $CBB, increment the address by $04 to page 2 and blink there at the right location before continuing the intercepted input normally through KEYIN ($FD1B).

Turning attention now to handling the switching between pages 1 and 2, we must put the proper addresses in the CSW and KSW locations. For page 2, lines $CD7-$CE9 put $C16 in CSW and $CB3 in KSW and put an identifying 2 on the lower right screen. For page 1, lines $CF5-$D03 restore COUT1($FDF0) for CSW and KEYIN($FD1B) for KSW.

In order to make the program compatible with DOS, the CSW and KSW intercept values in ISWITCH and P2SWITCH must be echoed into DOS memory by calling each time the values are switched the DOS routine whose address is at $3EA. If you don't have DOS, put EA in each byte in lines $CEC and $D05.

To trigger the switching process conveniently, we need some simple commands. I used 1 [Ctrl-Y] or 2 [Ctrl-Y] as commands in monitor to switch pages during program development. Now we really need a set of Applesoft commands which can be used from the keyboard or a program. Following many illustrations in recent issues of *Call -A.P.P.L.E.* we can turn readily to the Ampersand command (&) which initiates a JMP to whatever address is in $3F6 and $3F7 (Ref. Manual p.132). If we put $D09 there and define commands &T1 and &T2 for text pages 1 and 2, the A register has $54 in it for T when the jump is made to $D09 (Applesoft character values are $80 less than monitor values.)

This matches $D09's comparison CMP #$54. Then $D0D reads the next character using CHRGET $00B1 and checks for a one, $31, or a two, $32. If a one, addressing the screen soft switch ($C054) displays page 1, lines from $CF0 set the CSW and KSW addresses, and $FB39 (SETTXT) is a monitor routine for resetting a normal text window. If a two, $C055 displays page 2, lines from $CD2 set CSW, KSW and P2IDENT, and, skipping $D37-$D40, set a full text window and jump to P2VTAB $C00 to put the cursor on Page 2.

For a page 2 HOME command we can use &O (note: not a zero) which will be recognized at $D57 and quickly activates $C77, our P2-HOME routine. Note that &T2 acts like a TEXT command for page 2.

If we remember that the mixed Lo-Res graphics and text mode is displayable on both pages, we can establish commands &G1 and &G2 with the G identified at $D61, the two at $D68, and the one at $D7F. In both cases the subroutine to $D37 sets the text window from line $14 down. Note that the List 2 program handles page 2 text in both Lo-Res and Hi-Res mixed graphics formats with considerable ease. Unfortunately, in the Lo-Res case, the graphics cannot yet be entered directly on page 2.

Full Lo-Res graphics capability is possible with combined lists 2 and 3, but you may not need it. To use list 2 just for page 2 text, including the mixed formats, lines $CD2, $CD4, $CF0, $CF2 and $D3D must be negated by inserting EA (no opera-

tion) for each byte of code. Also you must change byte $D64 from 34 to 1E and move the initialization routine starting at $E8B to start instead at $D99. You can now set BASPRG to $E00 to allow Applesoft programs more RAM memory.

Although the page 2 text is now in quite serviceable form, the Lo-Res graphics is not. Unfortunately it depends even more directly on the analogous GBASCALC ($F847) in the monitor which also insists on page 1 addresses. In addition, the graphics, like the monitor scroll and home routines, does not go through the CSW intercept, so we must again move some monitor routines to RAM and modify them for page 2 use. Thus in List 3 starting at $E07, the graphics routines from the monitor (starting at $FB00) are largely reproduced so that they call on a GBASCALC ($E4E) which can be easily set for either page 1 or 2.

Additional & commands for the standard Lo-Res graphics inputs are defined and combined with interpreters in the range $D99-$E06. The List 2 program must be unchanged so that the page control switches at $CD2 and $CF0 include the switch in GBASCALC between $04 and $08 at $E52. In addition the &G1 and &G2 commands can include clearing the graphics by including subroutine $E3D at $D3D. The jump at $D63 must now go to the next command at $D99 rather than to the $F206 error message.

Note that the & graphics input routines can be used equally well on either page 1 or 2 and can handle "aexpr" forms for the arguments. Thus Applesoft PLOT X,Y becomes &P X,Y where X and Y can each be an aexpr. Beginning at $D99 the P is identified and CHRGET $00B1 moves TXTPTR $B8,9 to look at X. (Applesoft routines are outlined in *Call -A.P.P.L.E. in Depth #1* and *Apple Orchard Vol.1, #1.)* Calling PLOTFNS ($F1EC) ends up with the value of X in $F0 and of Y in $2C and $2D. A jump to PLOT $E07 puts the point in memory and thus on the screen as determined by GBASCALC($E4E). Similarly HLIN X1,X2 at Y becomes &H X1,X2 at Y, VLIN becomes &V Y1,Y2 at X and COLOR = N becomes &C N.

The SCRN(X,Y) command is more difficult to convert since it returns a number when called and an & command is not permitted to do this as directly. Thus the corresponding command is defined as &S X,Y,SC where SC defines a variable whose value is the return desired. You can use any other variable designation in place of SC in the command, if you wish.

The S is identified at $DE6, the values of the expressions for X and Y are handled by PLOTFNS at $DF0, recourse is made to SCRN at $E7A which returns the value desired in the A register. FLOAT ($EB93) puts this value in floating point form. CHKCOM ($DEBE) checks for the comma in the command and PTRGET ($DFE3) locates the address of the variable SC (or creates one if needed). $EB2B stores the value of SC allowing it to be called or used as needed in terms of the variable SC itself (or other designation you may have used).

The last section from $E8B is a short initialization routine which, when E8EG is entered in monitor mode or a CALL 3726 in Applesoft right after loading List 3, initializes the program by putting the JMP $D09 in the & location and establishes $F00 or 3840 as the beginning of Applesoft programs. Thus three zeroes are entered at $F00,1,2 and the high byte of the program beginning and end pointers is set to $0F. The routine SCRATCH ($D64B) from Applesoft resets the other pointers and various flags.

To permit the loading from disk in the BRUN mode we should also add a line just before $C00 which calls the INIT routine as:

BFD- 4C 8E 0E JMP $0E8E

This small part of the page 2 display range $800-$BFF does not actually correspond to screen positions. The entire program thus has address and length A$BFD and L$2AB when BSAVED. For those who have an Ap.LCase chip for lower case output and use the KEYPATCH routine, it is quite possible to have lower case on page 2 as well as page 1. Simply replace KEYIN ($FD1B) with $9C28, the KEYPATCH intercept location, in lines $CCF,$CFD and $D01.

I hope this development proves helpful and encouraging to others getting started in assembly language programming using monitor and Applesoft routines whenever appropriate. Note that many of the subroutines in Lists 2 and 3 can be called directly from the monitor or with an Applesoft CALL using the addresses in Table 1, located at end of listing 2 & 3. Since they are quite similar to those in the monitor, especially for Lo-Res graphics, the detailed information in *Apple][Monitors Peeled* (pp. 34-38 and pp. 68-69) can be quite helpful in understanding their use. Note also that you can readily augment or modify these List 2 and 3 programs to include & commands which meet your special needs.

```
:ASM

1        Combined listing 2 & 3
2
         ********************************
4        *                              *
5        *  Printing Text to Page Two   *
6        *                              *
7        *      by Bob Halfman          *
8        *                              *
9        * Call -A.P.P.L.E. : Jan. 1983 *
10       *                              *
         ********************************
12
13       *
14                ORG   $BFD
15       *
16       WNDLFT    EQU   $20
17       WNDWDTH   EQU   $21
18       WNDTOP    EQU   $22
19       WNDBTM    EQU   $23
20       CH        EQU   $24
21       CV        EQU   $25
22       GBASL     EQU   $26
23       GBASH     EQU   $27
24       BASL      EQU   $28
25       BASH      EQU   $29
26       BAS2L     EQU   $2A
27       BAS2H     EQU   $2B
28       H2        EQU   $2C
29       V2        EQU   $2D
```

DIRECT TO PAGE TWO WITH APPLESOFT

```
                        30   MASK      EQU   $2E
                        31   COLOR     EQU   $30
                        32   YSAV1     EQU   $35
                        33   CSWL      EQU   $36
                        34   CSWH      EQU   $37
                        35   KSWL      EQU   $38
                        36   KSWH      EQU   $39
                        37   TXTTABH   EQU   $68
                        38   PRGENDH   EQU   $B0
                        39   CHRGET    EQU   $B1
                        40   FIRST     EQU   $F0
                        41   DOSINT    EQU   $3EA
                        42   AMPJMP    EQU   $3F8
                        43   BASPRG    EQU   $0F00
                        44   KBD       EQU   $C000
                        45   KBDSTRB   EQU   $C010
                        46   TXTCLR    EQU   $C050
                        47   TXTSET    EQU   $C051
                        48   MIXSET    EQU   $C053
                        49   LOWSCR    EQU   $C054
                        50   HISCR     EQU   $C055
                        51   LORES     EQU   $C056
                        52   SCRTCH    EQU   $D64B
                        53   CHKCOM    EQU   $DEBE
                        54   PTRGET    EQU   $DFE3
                        55   GETBYT    EQU   $E6F8
                        56   MOVMF     EQU   $EB2B
                        57   FLOAT     EQU   $EB93
                        58   PLOTFNS   EQU   $F1EC
                        59   SYNERR    EQU   $F206
                        60   PLOTLIN   EQU   $F209
                        61   SETTXT    EQU   $FB39
                        62   BASCALC   EQU   $FBC1
                        63   KEYIN     EQU   $FD1B
                        64   COUT1     EQU   $FDF0
                        65
                        66   *
                        67
0BFD: 4C 8E 0E          68             JMP   INIT
0C00: A5 25             69   P2VTAB    LDA   CV
0C02: 20 C1 FB          70   P2VTABZ   JSR   BASCALC
0C05: 65 20             71             ADC   WNDLFT
0C07: 85 28             72             STA   BASL
0C09: 48                73   PAGE2     PHA
0C0A: A5 29             74             LDA   BASH
0C0C: C9 08             75             CMP   #$08
0C0E: B0 04             76             BCS   ON2
0C10: 69 04             77             ADC   #$04
0C12: 85 29             78             STA   BASH
0C14: 68                79   ON2       PLA
0C15: 60                80             RTS
                        81   *
0C16: 20 09 0C          82   INPUT     JSR   PAGE2
0C19: 84 35             83             STY   YSAV1
0C1B: A4 25             84             LDY   CV
0C1D: C8                85             INY
0C1E: C4 23             86             CPY   WNDBTM
0C20: 90 08             87             BCC   OUTPUT
0C22: A4 24             88             LDY   CH
0C24: C8                89             INY
0C25: C4 21             90             CPY   WNDWDTH
0C27: B0 09             91             BCS   EOBL
0C29: C9 8D             92             CMP   #$8D
0C2B: F0 71             93             BEQ   P2VIDWAIT
0C2D: A4 35             94   OUTPUT    LDY   YSAV1
0C2F: 4C F0 FD          95             JMP   COUT1
0C32: A8                96   EOBL      TAY
0C33: 10 04             97             BPL   FLINCH
0C35: C9 A0             98             CMP   #$A0
0C37: 90 04             99             BCC   P2SCROLL
0C39: A4 24            100   FLINCH    LDY   CH
0C3B: 91 28            101             STA   (BASL),Y
0C3D: A9 00            102   P2SCROLL  LDA   #$00
0C3F: 85 24            103             STA   CH
0C41: A5 22            104             LDA   WNDTOP
0C43: 48               105             PHA
0C44: 20 02 0C         106             JSR   P2VTABZ
0C47: A5 28            107   P2SCROLL1 LDA   BASL
0C49: 85 2A            108             STA   BAS2L
0C4B: A5 29            109             LDA   BASH
0C4D: 85 2B            110             STA   BAS2H
0C4F: A4 21            111             LDY   WNDWDTH
0C51: 88               112             DEY
0C52: 68               113             PLA
0C53: 69 01            114             ADC   #$01
0C55: C5 23            115             CMP   WNDBTM
0C57: B0 0D            116             BCS   P2SCROLL3
0C59: 48               117             PHA
0C5A: 20 02 0C         118             JSR   P2VTABZ
0C5D: B1 28            119   P2SCROLL2 LDA   (BASL),Y
0C5F: 91 2A            120             STA   (BAS2L),Y
0C61: 88               121             DEY
0C62: 10 F9            122             BPL   P2SCROLL2
0C64: 30 E1            123             BMI   P2SCROLL1
0C66: A0 00            124   P2SCROLL3 LDY   #$00
0C68: 20 6D 0C         125             JSR   P2CLEOLZ
0C6B: B0 24            126             BCS   P2VTAB1
0C6D: A9 A0            127   P2CLEOLZ  LDA   #$A0
0C6F: 91 28            128   P2CLEOL2  STA   (BASL),Y
0C71: C8               129             INY
0C72: C4 21            130             CPY   WNDWDTH
0C74: 90 F9            131             BCC   P2CLEOL2
0C76: 60               132             RTS
                       133   *
0C77: 84 35            134   P2HOME    STY   YSAV1
0C79: A5 22            135             LDA   WNDTOP
0C7B: 85 25            136             STA   CV
0C7D: A0 00            137             LDY   #$00
0C7F: 84 24            138             STY   CH
0C81: 48               139   P2CLEOP1  PHA
0C82: 20 02 0C         140             JSR   P2VTABZ
0C85: 20 6D 0C         141             JSR   P2CLEOLZ
0C88: A0 00            142             LDY   #$00
0C8A: 68               143             PLA
0C8B: 69 00            144             ADC   #$00
0C8D: C5 23            145             CMP   WNDBTM
0C8F: 90 F0            146             BCC   P2CLEOP1
0C91: 20 00 0C         147   P2VTAB1   JSR   P2VTAB
0C94: 48               148             PHA
0C95: A9 B2            149             LDA   #$B2
0C97: 8D F7 0B         150             STA   $0BF7
0C9A: 68               151             PLA
0C9B: A4 35            152             LDY   YSAV1
0C9D: 60               153             RTS
0C9E: AC 00 C0         154   P2VIDWAITLDY   KBD
0CA1: C0 93            155             CPY   #$93
0CA3: D0 0B            156             BNE   NOWAIT
0CA5: 2C 10 C0         157             BIT   KBDSTRB
0CA8: AC 00 C0         158   P2KBDWAITLDY   KBD
0CAB: 10 FB            159             BPL   P2KBDWAIT
0CAD: 2C 10 C0         160             BIT   KBDSTRB
0CB0: 4C 3D 0C         161   NOWAIT    JMP   P2SCROLL
0CB3: 48               162   P2EDIT    PHA
0CB4: A5 29            163             LDA   BASH
0CB6: C9 08            164             CMP   #$08
0CB8: 68               165             PLA
0CB9: B0 14            166             BCS   NOEDIT
0CBB: 91 28            167             STA   (BASL),Y
0CBD: E6 29            168             INC   BASH
0CBF: E6 29            169             INC   BASH
0CC1: E6 29            170             INC   BASH
0CC3: E6 29            171             INC   BASH
0CC5: B1 28            172             LDA   (BASL),Y
0CC7: 48               173             PHA
0CC8: 29 3F            174             AND   #$3F
0CCA: 09 40            175             ORA   #$40
0CCC: 91 28            176             STA   (BASL),Y
0CCE: 68               177             PLA
0CCF: 4C 1B FD         178   NOEDIT    JMP   KEYIN
0CD2: A9 08            179   P2SWITCH  LDA   #$08
0CD4: 8D 53 0E         180             STA   GBASCALC+5
0CD7: A9 16            181             LDA   #<INPUT
```

```
0CD9: 85 36      182          STA  CSWL
0CDB: A9 0C      183          LDA  #>INPUT
0CDD: 85 37      184          STA  CSWH
0CDF: A9 B3      185          LDA  #<P2EDIT
0CE1: 85 38      186          STA  KSWL
0CE3: A9 0C      187          LDA  #>P2EDIT
0CE5: 85 39      188          STA  KSWH
0CE7: A9 B2      189          LDA  #$B2
0CE9: 8D F7 0B   190          STA  $0BF7
0CEC: 20 EA 03   191          JSR  DOSINT
0CEF: 60         192          RTS
                 193  *
0CF0: A9 04      194  ISWITCH  LDA  #$04
0CF2: 8D 53 0E   195          STA  GBASCALC+5
0CF5: A9 F0      196          LDA  #<COUT1
0CF7: 85 36      197          STA  CSWL
0CF9: A9 FD      198          LDA  #>COUT1
0CFB: 85 37      199          STA  CSWH
0CFD: A9 1B      200          LDA  #<KEYIN
0CFF: 85 38      201          STA  KSWL
0D01: A9 FD      202          LDA  #>KEYIN
0D03: 85 39      203          STA  KSWH
0D05: 20 EA 03   204          JSR  DOSINT
0D08: 60         205          RTS
                 206  *
0D09: C9 54      207  AMPT     CMP  #$54
0D0B: D0 4A      208          BNE  AMPO
0D0D: 20 B1 00   209          JSR  CHRGET
0D10: C9 31      210          CMP  #$31
0D12: D0 0C      211          BNE  AMPT2
0D14: 20 B1 00   212          JSR  CHRGET
0D17: AD 54 C0   213  ITEXT    LDA  LOWSCR
0D1A: 20 F0 0C   214          JSR  ISWITCH
0D1D: 4C 39 FB   215          JMP  SETTXT
0D20: C9 32      216  AMPT2    CMP  #$32
0D22: F0 03      217          BEQ  NOERR
0D24: 4C 06 F2   218          JMP  SYNERR
0D27: 20 B1 00   219  NOERR    JSR  CHRGET
0D2A: AD 55 C0   220  P2TEXT   LDA  HISCR
0D2D: 20 D2 0C   221          JSR  P2SWITCH
0D30: AD 51 C0   222  P2SETTXT LDA  TXTSET
0D33: A9 00      223          LDA  #$00
0D35: F0 0B      224          BEQ  SETWND
0D37: AD 50 C0   225  P2SETGR  LDA  TXTCLR
0D3A: AD 53 C0   226          LDA  MIXSET
0D3D: 20 3D 0E   227          JSR  CLRTOP
0D40: A9 14      228          LDA  #$14
0D42: 85 22      229  SETWND   STA  WNDTOP
0D44: A9 00      230          LDA  #$00
0D46: 85 20      231          STA  WNDLFT
0D48: A9 28      232          LDA  #$28
0D4A: 85 21      233          STA  WNDWDTH
0D4C: A9 18      234          LDA  #$18
0D4E: 85 23      235          STA  WNDBTM
0D50: A9 17      236          LDA  #$17
0D52: 85 25      237  P2TABV   STA  CV
0D54: 4C 00 0C   238          JMP  P2VTAB
0D57: C9 4F      239  AMPO     CMP  #$4F
0D59: D0 06      240          BNE  AMPG
0D5B: 20 B1 00   241          JSR  CHRGET
0D5E: 4C 77 0C   242          JMP  P2HOME
0D61: C9 47      243  AMPG     CMP  #$47
0D63: D0 34      244          BNE  AMPPXY
0D65: 20 B1 00   245          JSR  CHRGET
0D68: C9 32      246          CMP  #$32
0D6A: D0 13      247          BNE  AMPG1
0D6C: 20 B1 00   248          JSR  CHRGET
0D6F: AD 56 C0   249          LDA  LORES
0D72: AD 53 C0   250          LDA  MIXSET
0D75: 20 D2 0C   251          JSR  P2SWITCH
0D78: 20 37 0D   252          JSR  P2SETGR
0D7B: AD 55 C0   253          LDA  HISCR
0D7E: 60         254          RTS
                 255  *
0D7F: C9 31      256  AMPG1    CMP  #$31
0D81: F0 03      257          BEQ  NOERR1
0D83: 4C 06 F2   258  ERR      JMP  SYNERR
0D86: 20 B1 00   259  NOERR1   JSR  CHRGET
0D89: AD 56 C0   260          LDA  LORES
0D8C: AD 53 C0   261          LDA  MIXSET
0D8F: 20 F0 0C   262          JSR  ISWITCH
0D92: 20 37 0D   263          JSR  P2SETGR
0D95: AD 54 C0   264          LDA  LOWSCR
0D98: 60         265          RTS
                 266  *
0D99: C9 50      267  AMPPXY   CMP  #$50
0D9B: D0 10      268          BNE  AMPH
0D9D: 20 B1 00   269          JSR  CHRGET
0DA0: 20 EC F1   270          JSR  PLOTFNS
0DA3: 8A         271          TXA
0DA4: A4 F0      272          LDY  FIRST
0DA6: C0 28      273          CPY  #$28
0DA8: B0 D9      274  ERR2     BCS  ERR
0DAA: 4C 07 0E   275          JMP  PLOT
0DAD: C9 48      276  AMPH     CMP  #$48
0DAF: D0 12      277          BNE  AMPV
0DB1: 20 B1 00   278          JSR  CHRGET
0DB4: 20 09 F2   279          JSR  PLOTLIN
0DB7: 8A         280          TXA
0DB8: A4 2C      281          LDY  H2
0DBA: C0 28      282          CPY  #$28
0DBC: B0 EA      283          BCS  ERR2
0DBE: A4 F0      284          LDY  FIRST
0DC0: 4C 20 0E   285          JMP  HLINE
0DC3: C9 56      286  AMPV     CMP  #$56
0DC5: D0 11      287          BNE  AMPC
0DC7: 20 B1 00   288          JSR  CHRGET
0DCA: 20 09 F2   289          JSR  PLOTLIN
0DCD: 8A         290          TXA
0DCE: A8         291          TAY
0DCF: C0 28      292          CPY  #$28
0DD1: B0 D5      293          BCS  ERR2
0DD3: A5 F0      294          LDA  FIRST
0DD5: 4C 2F 0E   295          JMP  VLINE
0DD8: C9 43      296  AMPC     CMP  #$43
0DDA: D0 0A      297          BNE  AMPS
0DDC: 20 B1 00   298          JSR  CHRGET
0DDF: 20 F8 E6   299          JSR  GETBYT
0DE2: 8A         300          TXA
0DE3: 4C 6D 0E   301          JMP  SETCOL
0DE6: C9 53      302  AMPS     CMP  #$53
0DE8: F0 03      303          BEQ  NOERR3
0DEA: 4C 06 F2   304          JMP  SYNERR
0DED: 20 B1 00   305  NOERR3   JSR  CHRGET
0DF0: 20 EC F1   306          JSR  PLOTFNS
0DF3: 8A         307          TXA
0DF4: A4 F0      308          LDY  FIRST
0DF6: 20 7A 0E   309          JSR  SCRN
0DF9: 20 93 EB   310          JSR  FLOAT
0DFC: 20 BE DE   311          JSR  CHKCOM
0DFF: 20 E3 DF   312          JSR  PTRGET
0E02: AA         313          TAX
0E03: 20 2B EB   314          JSR  MOVMF
0E06: 60         315          RTS
                 316  *
                 317  *
                 318  *
0E07: 4A         319  PLOT     LSR
0E08: 08         320          PHP
0E09: 20 4E 0E   321          JSR  GBASCALC
0E0C: 28         322          PLP
0E0D: A9 0F      323          LDA  #$0F
0E0F: 90 02      324          BCC  RTMASK
0E11: 69 E0      325          ADC  #$E0
0E13: 85 2E      326  RTMASK   STA  MASK
0E15: B1 26      327  PLOT1    LDA  (GBASL),Y
0E17: 45 30      328          EOR  COLOR
0E19: 25 2E      329          AND  MASK
0E1B: 51 26      330          EOR  (GBASL),Y
0E1D: 91 26      331          STA  (GBASL),Y
0E1F: 60         332          RTS
                 333  *
0E20: 20 07 0E   334  HLINE    JSR  PLOT
0E23: C4 2C      335  HLINE1   CPY  H2
0E25: B0 11      336          BCS  RTS1
0E27: C8         337          INY
0E28: 20 15 0E   338          JSR  PLOT1
0E2B: 90 F6      339          BCC  HLINE1
```

DIRECT TO PAGE TWO WITH APPLESOFT

```
0E2D: 69 01      340 VLINEZ   ADC  #$01
0E2F: 48         341 VLINE    PHA
0E30: 20 07 0E   342          JSR  PLOT
0E33: 68         343          PLA
0E34: C5 2D      344          CMP  V2
0E36: 90 F5      345          BCC  VLINEZ
0E38: 60         346 RTS1     RTS
                 347 *
0E39: A0 2F      348 CLRSCR   LDY  #$2F
0E3B: D0 02      349          BNE  CLRSCR2
0E3D: A0 27      350 CLRTOP   LDY  #$27
0E3F: 84 2D      351 CLRSCR2  STY  V2
0E41: A0 27      352          LDY  #$27
0E43: A9 00      353 CLRSCR3  LDA  #$00
0E45: 85 30      354          STA  COLOR
0E47: 20 2F 0E   355          JSR  VLINE
0E4A: 88         356          DEY
0E4B: 10 F6      357          BPL  CLRSCR3
0E4D: 60         358          RTS
                 359 *
0E4E: 48         360 GBASCALC PHA
0E4F: 4A         361          LSR
0E50: 29 03      362          AND  #$03
0E52: 09 04      363          ORA  #$04
0E54: 85 27      364          STA  GBASH
0E56: 68         365          PLA
0E57: 29 18      366          AND  #$18
0E59: 90 02      367          BCC  GBCALC
0E5B: 69 7F      368          ADC  #$7F
0E5D: 85 26      369 GBCALC   STA  GBASL
0E5F: 06 26      370          ASL  GBASL
0E61: 06 26      371          ASL  GBASL
0E63: 05 26      372          ORA  GBASL
0E65: 85 26      373          STA  GBASL
0E67: 60         374          RTS
                 375 *
0E68: A5 30      376          LDA  COLOR
0E6A: 18         377          CLC
0E6B: 69 03      378          ADC  #$03
0E6D: 29 0F      379 SETCOL   AND  #$0F
0E6F: 85 30      380          STA  COLOR
0E71: 0A         381          ASL
0E72: 0A         382          ASL
0E73: 0A         383          ASL
0E74: 0A         384          ASL
0E75: 05 30      385          ORA  COLOR
0E77: 85 30      386          STA  COLOR
0E79: 60         387          RTS
                 388 *
0E7A: 4A         389 SCRN     LSR
0E7B: 08         390          PHP
0E7C: 20 4E 0E   391          JSR  GBASCALC
0E7F: B1 26      392          LDA  (GBASL),Y
0E81: 28         393          PLP
0E82: 90 04      394          BCC  RTMSKZ
0E84: 4A         395          LSR
0E85: 4A         396          LSR
0E86: 4A         397          LSR
0E87: 4A         398          LSR
0E88: 29 0F      399 RTMSKZ   AND  #$0F
0E8A: 60         400          RTS
                 401 *
0E8B: 4C 09 0D   402          JMP  AMPT
0E8E: A2 03      403 INIT     LDX  #$03
0E90: BD 8A 0E   404 INIT1    LDA  INIT-4,X
0E93: 9D F4 03   405          STA  AMPJMP-4,X
0E96: A9 00      406          LDA  #$00
0E98: 9D FF 0E   407          STA  BASPRG-1,X
0E9B: CA         408          DEX
0E9C: D0 F2      409          BNE  INIT1
0E9E: A9 0F      410          LDA  #>BASPRG
0EA0: 85 68      411          STA  TXTTABH
0EA2: 85 B0      412          STA  PRGENDH
0EA4: 20 4B D6   413          JSR  SCRTCH
0EA7: 60         414          RTS

--End assembly--

683 bytes

Errors: 0
```

Function	Hex Addr	Dec Addr	Program Label	Regs Destr
Set Page 2 BASL,H from CV and WNDLFT	C00	3072	P2VTAB	A
Set Page 2 BASL,H from A-reg,WNDLFT	C02	3074	P2VTABZ	A
Ensure Page 2, saving all registers	C09	3081	PAGE2	None
Character print to Page 2 screen output - Page 2 version of COUT1	C16	3094	INPUT	?A
Scroll Page 2 window	C3D	3133	P2SCROLL	A,Y
Clear Page 2 window,cursor top left	C77	3191	P2HOME	A
Set CSW,KSW,GBASCALC,P2IDENT to Page 2	CD2	3282	P2SWITCH	A
Put 2 on Page 2 screen lower right	CE7	3303	P2IDENT	A
Set CSW,KSW,GBASCALC to Page 1	CF0	3312	ISWITCH	A
Same as &T1 - set Page 1 display, ISWITCH, full text window	D17	3351	ITEXT	A
Same as &T2 - set Page 2 display P2SWITCH,P2SETTXT	D2A	3370	P2TEXT	A
Set full text wndw,cursor bttm Page 2	D30	3376	P2SETTXT	A
Set mixed graphics,clear LORES window	D37	3383	P2SETGR	A
Set full text wndw,excpt top from A-reg	D42	3394	P2SETWNDW	A
Place cursor at line A, column CH	D52	3410	P2TABV	A
Same as &G1 - set mixed LORES, clear graphics, ISWITCH	D89	3465	IGR	A
Same as &G2 - set mixed LORES, clear graphics, P2SWITCH	D6F	3439	P2GR	A
Clear full graphics screen Page 1,2	E39	3641	CLRSCR	A,Y
Clear mixed graphics screen Page 1,2	E3D	3645	CLRTOP	A,Y

TABLE 1

APPLEMASH:

GETTING THE POINT

Michael Thyng

LET'S take care of a little unfinished business. I was taken to task for my remarks about GET. Kay from Pennsylvania wrote a rather long rebuttle. She makes such good points that I want to include her letter in its entirety. I'll comment at the end.

"Dear Val,

I've often admired Michael Thyng's talent for explaining Applesoft in plain English in his "Applemash" columns. It seems to me, though, that he may have sold GET short in his June, 1982, piece called "GETting Straight." There's no doubt that the budding programmer can GET more.

Most readers have probably seen the forms data entry programs using GET in the Apple II User's Guide. That's not so dissimilar from word processing, I suppose, so it may not represent a real addition to Michael's list of GET uses. But what about using GET to read a "mystery" text file character by character? This process is discussed on page 54 of the DOS Manual and on page 184 of the User's Guide. Of course, now that we have that handy "FILEDUMP" utility from the June, 1982, Call–A.P.P.L.E., *this painful method isn't needed for sequential files any more. Nevertheless, it exists, and suppose I've left my disk with FILEDUMP on it at home?*

The most useful application of GET that Michael didn't mention, however, is in the area of menu speed–up. When a programmer is ready to attempt a menu–driven program, there will be many cases when a single–character response is desired. Why waste time with the carriage return required for INPUT, when using GET zips the choice process along faster? In addition, avoiding the carriage return keeps the inexperienced user's fingers away from the perilous RESET key. I've used and written menus both ways, and the increased ease and speed of GET is remarkable.

What if you have more than nine menu choices? Can you still use GET without using a carriage return to indicate the entry is finished? Quite often you can. You can use letters instead of or in addition to the number codes in the menu. Figuring out how to error trap that will be good exercise! You might even realize that your menu is too complicated and that you need to break things down into submenus.

If you have a choice point that amounts to a "stop now" versus "go ahead", GETting the response allows you to use the escape key as a stop signal. You can't INPUT escape. Using GET is the way to enable escape, located away from RESET and not familiar enough to many users to be confused with something else, as a way "out" of a process. Granted, if you only want to pause program operation, not get some code from the keyboard, GET produces too much garbage. If you keep using a different variable name every time you GET a menu response, more garbage. Then solution is to re–use variable names (KEY$, DO$, CODE$, etc.) at menus when you won't need the value forever. If you only want to pause, you can use something other than GET. The possibilities include CALL – 756, PEEK (–16384), and WAIT. These will call a temporary halt while the user reads some instructions, for example, at his own pace. By the time someone is ready to write a long, complex program using many strings, he is also ready to learn about forcing garbage collection with the X = FRE(0) command.

There are also, admittedly, several ways to GET exasperated. Those of us who read the instructions only when all else fails are especially vulnerable to the pitfalls of GET (sounds like an adventure game?!). If you use a GET immediately before a DOS command within a program, and if you haven't paid attention to good old page 54 of the DOS manual, the program will undoubtedly fail. DOS commands will not work unless preceded by a carriage return (CHR$(13)), and GET doesn't provide one — unlike INPUT. Files won't open, printers won't turn on, catalogs won't appear; all you get is the command printed on screen, and a case of frustration. The solution, as the manual plainly tells us, is to put in a CHR$(13) before each DOS command. You can set D$ equal to CHR$(13) + CHR$(4), and then DOS does when the GET is gotten.

If a program has one or more submenus following a main menu, all using GETs, another problem can arise. If your distracted or hamfisted user presses two keys at the main menu, the program will probably accept the second keypress as the response to the submenu. Who knows what that will cause? You can sometimes prevent this by error trapping and by using mutually–exclusive codes in the menus. For example, you could use numbers at the main menu and letters at the submenus; then your error trap would reject any number that tried to get past the submenu from a goofed menu response. Of course if the pause following the menu isn't a GET but something like CALL – 756, you can't save the day with a trap. You've saved on garbage but created a risk for seriously klutzy users.

In summary, there are several uses of GET that Michael did not cover:

1. *data entry*
2. *reading "mystery" files*
3. *menu speed–up*
4. *allowing escape as an "out"*

There are some disadvantages, as I've pointed out, but many of them can be worked around as the programmer gains skill and reads more about the Apple. One thing I like about Call – A.P.P.L.E. *is its attention to the beginner — to all levels of programmers really, so I thought a few words on GET might be worthwhile.*

Kay

I can't argue with logic. What Kay says is true and she makes good points. Experienced programmers should use GET in the appropriate application. However... the purpose of Applemash is to explain in detail the way things work. I have geared my remarks primarily to the less experienced and beginning programmers. Occasionally we discuss things with a bit more complexity that the experienced programmers are ready for, but "Mash" is designed as information for beginners.

Now let's (should I pun it) GET into this month's material. We started Lo–Res graphics in November

—●→

GETTING THE POINT

and we want to explore it a little more in depth. I promised you graphing using data stored on a disk file, data scaling, and a nifty little Lo-Res drawing program.

First, we'll put data in a text file on a diskette. For simplicity, we'll use the random number generator to accumulate some data, and a sequential text file to record it to disk.

Enter the following program and save it.

```
]LIST

5  REM EXAMPLE 1
10  REM   CREATE RANDOM
      COUNT FOR SEQUENTIAL
      FILE
100 D$ =  CHR$ (4)
200 F1$ = "TRAFFIC COUNT"
300 X = 1
400  PRINT D$;"OPEN";F1$
500  REM FETCH RANDOM
      NUMBER
600 X% =  RND (8) *
      10000
700  PRINT D$;"WRITE";F1$
800  PRINT X%
900  PRINT D$
1000 X = X + 1
1100  IF X < 25 THEN
      500
1200  PRINT D$;"WRITE";F1$
1300  PRINT 9999
1400  PRINT D$
1500  PRINT D$;"CLOSE"
1600  PRINT "PROGRAM
      IS COMPLETE"
```

Now run the program. Okay, now you have 24 pieces of data stored on diskette. Let's assume the data is a count of the number of cars passing a certain point on Interstate 5. *Note: Your data will be different from mine.* After all, why do you think they call it a RANDOM number generator. In fact, if you run this ten times, you will get ten different sets of 24 values. For consistency and understanding, I will refer to the data that my programs have generated. When I ran the program, I got the following 24 values.

```
]LIST

5  REM EXAMPLE 2
10  REM   READ TRAFFIC
      COUNT DATA AND
      GRAPH IT
20  DIM C%(24)
30 LOW% = 9999:HI% =
      0
40 X = 1
100 D$ =  CHR$ (4)
200 F1$ = "TRAFFIC COUNT"
300  PRINT D$;"OPEN";F1$
400  REM FETCH TRAFFIC
      COUNT
500  PRINT D$;"READ";F1$
600  INPUT X%
700  PRINT D$
800  IF X% = 9999 THEN
      1500
900 C%(X) = X%
1000 X = X + 1
1100  PRINT X%
1200  IF X% > HI% THEN
      HI% = X%
1300  IF X% < LOW% THEN
      LOW% = X%
1400  GOTO 500
1500  PRINT D$;"CLOSE"
1600  PRINT "LOW = ";LOW%
1700  PRINT "HIGH =
      ";HI%
1800  REM SCALING AREA
1900 SPRED% = HI% -
      LOW%
2000  PRINT SPRED%
2100 UNIT% = SPRED%
      / 40
2200  PRINT UNIT%
2300  FOR N = 1 TO 24
2400  PRINT C%(N);"
      ";
2500 C%(N) = C%(N) /
      UNIT%
2600  PRINT C%(N)
2700  NEXT N
2800  REM GRAPH THE
      DATA
2900  GR : COLOR= 9
3000  FOR N = 1 TO 24
3100  FOR F = 39 TO
      (40 - C%(N)) STEP
      - 1
3200  PLOT N + 5,F
3300  NEXT F
3400  NEXT N
3500  PRINT "HOUR  123456789
      1111111111122222"
3600  PRINT "COUNTS
                01234567890123
      4"
```

Now let's do something called scaling. What we want to do is graph our data onto a screen that has 40 horizontal and 40 vertical plotting positions (remember this is Lo-Res.) We used the random number generator to provide us with 35 values. After we read the values in from the text file, we want to plot them as we move across the horizontal axis. We don't have to scale in this direction.

The horizontal axis will be the days that the count was taken. The count of each day will be plotted vertically. Now look at the data. The values range from a low of 23 to a high of 8932. These are larger than 40. How can we plot them on our screen? Well, we do it by scaling.

Scaling is the technique of converting our data into representative blocks of data that are less (in this case) than our original data but maintain the relationship of the data. We divide the data into 40 equal or nearly equal parts. Then we represent each plotted square as the count of a certain number of vehicles per hour. The graph of our data should look something like the ones we worked on before.

In line 1800 we subtract the low value from the high value to establish the range of values we have to scale. Then we divide it by 40. This gives us a scale of 223 cars counted per square plotted on our graph. Remember that a graph on a screen is not intended to give your audience precise figures, only a quick visual reference. If you want precision, then you print out the numeric values in a chart or matrix.

The other thing I promised you last time was a way to draw Lo-Res pictures on the screen without using paddles. Enter the program below and then I'll tell you what each part does and why.

```
]LIST

5  REM EXAMPLE 3
7  REM WRITTEN BY:
8  REM   MICHAEL THYNG
9  REM   SEPT. 22, 1982
10  REM DRAWING WITH
      LO-RES
100  HOME : PRINT
200  PRINT "ENTER BACKGROUND
      COLOR"
300  PRINT
400  PRINT "  0 FOR
      STANDARD"
500  INPUT " CHOOSE
      0 - 15  ";C
600  IF C > 15 THEN
      100
700  GR
800  COLOR= C
900  FOR H = 0 TO 19
1000  HLIN 0,39 AT H
1100  HLIN 0,39 AT (39
      - H)
1200  NEXT H
1300 C = 1
1400 H = 20:V = 20
1500 Z =  PEEK ( - 16384)
1600  POKE  - 16368,0
1700  IF Z = 201 THEN
      V = V - 1
1800  IF V < 0 THEN
      V = 0
1900  IF Z = 202 THEN
      H = H - 1
2000  IF H < 0 THEN
      H = 0
2100  IF Z = 204 THEN
      H = H + 1
2200  IF H > 39 THEN
      H = 39
2300  IF Z = 205 THEN
      V = V + 1
```

```
2400    IF V > 39 THEN
        V = 39
2500    IF Z = 195 THEN
        C = C + 1
2600    IF C > 15 THEN
        C = 0
2700    IF Z = 197 THEN
         TEXT : LIST :
        END
2800    IF Z = 211 THEN
        100
2900    COLOR= C
3000    PLOT H,V
3100    GOTO 1500
```

Lines 100 to 1200 let you choose a background color for the screen. Page 85 of my Applesoft II Basic Programming Reference Manual shows the colors available by number. (I trust you do realize you have to have a color TV or color monitor to get color output.) Line 1300 gives us a first plot color at location 20,20. Line 1400 holds our initial plot coordinates, H for horizontal and V for vertical. The "trick" comes in at lines 1500 to 2800. What we're doing is using a feature of the Apple that reads the keyboard without INPUT or GET. Line 1500 sets numeric variable Z equal to a number corresponding to the ASCII value of the key that was pressed. In English, that means when you press a J, numeric variable Z is set equal to 201. When you press a C, then Z is set equal to 195. I think you see the pattern. Now, look at line 1600. If that line was not there, Z would always be equal to the 201 (the ASCII value of the first key we pressed – J.) It resets the flag that allows a new value to be read from the keyboard. By the way, I used Z. I could have used any other valid numeric variable name.

When you run this program, you control directions with I, J, L, and M for up, left, right and down respectively. Pushing S clears the screen and C changes the color of the squares we are plotting. If you push E, the game is over, you are shifted back into text mode and the program is listed.

I wanted to accomplish two things with this drawing program. First, I wanted you to find out how to work without paddles. Second, I wanted to challenge you to make improvements to this program and send them to me. I'll pass them along to everybody else. HINT:

Wouldn't it be nice to save a good drawing on disk???

A.P.P.L.E. INFO (Continued from page 7)

It is important to be aware of time zone differences. If you reach the recorder late in the business day, try to also leave your home phone number. If you are calling from the east coast, and you call before noon (9 AM Pacific time), you are likely to reach our recorder.

Our local membership comprises only a small percentage of the total, thus the large majority of our calls are long distance. If you are asked to hold, we are most likely on the other line assisting another long distance caller with a problem. If requested, overseas calls will be given priority; all others will be handled *strictly in the order received.* Once we have taken your call, we will then be able to give you our undivided attention.

You can play a major role in improving the Hot Line efficiency and reducing waiting time for other members by preparing yourself beforehand with notes relating to your question or questions. For example, if you have a bug in a program, having a printout of it may prove helpful, or a list of possible causes for the error which you have already ruled out.

Some Hot Numbers

Questions relating to joining A.P.P.L.E., membership, orders, product information and shipping should be directed to the general office at (206) 271-4514. Call -A.P.P.L.E. display advertising rates and information are available from our advertising representatives, Grawin Publications, at (206) 223-0861.The Hot Line number itself remains (206) 932-6588

Another service of A.P.P.L.E. that goes hand-in-hand with the Hot Line is our A.P.P.L.E. Crate Bulletin Board, at (206) 935-9119, which can be accessed by members and readers with telecommunications capabilities. Here can be found, in addition to the expected message service, meeting notices, new product information, programming tips, and the A.P.P.L.E. Doctor, an alternative means of providing programming and technical assistance.

Each issue of Call —A.P.P.L.E. has a complete listing of names and home telephone numbers of club officers and directors, and technical consultants whose field of expertise is listed with the name. Our consultants are all unpaid volunteers who generously donate many hours to answering members' questions. Please give them every consideration. In particular we ask you check their area codes and know what *their* time is before placing the call. The Hot Line is equipped to answer most questions of a general nature, excluding Pascal, or can refer your question on. *Those questions of a highly technical nature, or on Pascal, might best be directed to one of our consultants.*

Since our consultants are volunteers, it is a reasonable assumption that most of the listed numbers are home numbers and should therefore be called in the evening.

We are constantly looking for qualified individuals who might like to serve as one of our technical consultants. For those who are interested, please drop a note in the mail to the editorial office, indicating your area(s) of interest and qualifications. Our consultants receive no remuneration, but their valuable service is the basis of all user groups: members help members. That's what our club is all about.

With that thought in mind, we would like to personally thank all of our callers for the courtesy they have shown in keeping their calls brief and to the point, and for their contributions to our pool of Apple information and knowledge.

1982 CALL —A.P.P.L.E. INDEX

Compiled by Kevin O. Donohoe and Val J. Golding

TITLE	AUTHOR	ISSUE	PAGE
ADVANCED TECHNICAL			
Applesoft's CHARGET Routine	Cornelis Bongers	Mar	21
Applesoft Internal Structure	C.K. Mesztenyi	Jan	9
Beware the $CFXX	David Goss	Aug	43
Executing Deferred Commands From the Keyboard	C. Dorsman	Aug	40
Expanding CHRGET	Anthony Rose	Sep	57
Getting the High Sine: Parsing String Formulas	Charles Kluepfel	Mar	27
IF/THEN/ELSE	Cornelis Bongers	Oct	57
Local Variables in Applesoft	Charles Kluepfel	Oct	13
Look Before You JMP	Ken Manley	May	83
Squashing A Nasty DOS Bug	Art Schumer	Aug	57
The $48 Trap	Cornelis Bongers	Jan	15
The Unknown INTFLG	Kelly W. Puckett	Dec	23
Unraveling An Applesoft Puzzle	Cornelis Bongers	Jul	45
APPLICATIONS			
Calendar Display Routine	Mike Firth	Jun	55
Christmas Present From a Hacker	Phillip Chien	Dec	63
Confessions of a Computer Junkie	Donal Buchanan	Nov	47
DICE 82	Dan Paymar	Apr	21
Disk/Tape Backup	Grant Schenk	Apr	9
Envelope Addresser	Ralph Swerdlow,M.D.	Oct	9
Game Paddle Printer Driver	Glen Bredon and Val J. Golding	Feb	51
Get Most Followup	Clay Ruth	Apr	71
Get the Most From a Diskette	Clay Ruth	Mar	13
Money Manager	Nelson Capes	Sep	21
MX 100 Hi-Res Dump	Bruce C. Detterich	Mar	41
Picture Compression	Edward C. So	May	21
Pitch & Rhythm on the Apple	Jack H. Bender	Jun	15
Playing Card Generator	Vincent Adereate	Nov	31
Random Adventures	David Sparks	May	37
Text File Fast Loader	Robert Purves	Sep	39
Text Screen Dump	David Walker	Feb	37
Total Recall	David Linker, M.D.	Aug	15
DOS			
A New DOS Command: Filedump	Lee Reynolds	Jun	47
Disk Verify	Roger Chaffee	Jun	39
DOS/Pascal Bilingual Disks	Mat Davis	Oct	17
DOS Patch for 80 Column Cards	Glen Bredon	Jul	75
Dual Boot	Bill Bush	Jun	49
Dual Drive Master Create	David L. Deever	Jul	65
Examining Directory Sectors	David Huffman	Jul	15
Free Space on the Screen	Mike Firth	Sep	47
Free Space Patch	Val J. Golding and Mark Pump	Apr	65
Get File Information Subroutine	Lou Infeld	Jan	19
Get Most Followup	Clay Ruth	Apr	71
Get the Most From a Diskette	Clay Ruth	Mar	13
Patching DOS the Easy Way	Leighton Paul	Jul	49
"Patching" the "Free Space Patch"	C.J. Welman	Jun	59
Squashing A Nasty DOS Bug	Art Schumer	Aug	57
Text File Fast Loader	Robert Purves	Sep	39
Yes, You Really Can Put files on Track 0	Clay Ruth	Dec	57
HI-RES GRAPHICS			
Champagne Graphics (on a beer budget)	Andrew L. Berentes	Nov	37
Character Generator ROM	Ian M. Jackson	Nov	21
Color Me Apple	M.A. Iannce	Nov	9
Confessions of a Computer Junkie	Donal Buchanan	Nov	47
Hi-Res Full Scroll	Edward C. So	Feb	23
Hi-Res Hi-Jinx	Edward C. So	Apr	59
MX 100 Hi-Res Dump	Bruce C. Detterich	Mar	41
Penguin Responds to Champagne Graphics		Nov	44
Playing Card Generator	Vincent Adereate	Nov	31
Slow Plot	Jim Morriset	Nov	63
REVIEWS			
Apple Pascal A Hands-On Approach	Lee A. Webster	Sep	45
BASIS 108 . . . An Alternative	Cornelis Bongers	Aug	23
Champagne Graphics (on a beer budget)	Andrew L. Berentes	Nov	37
Global P.L.E. Review	M. Joseph Schaller	Oct	82
Mae Macro-Assembler: Eastern House Responds	Don Earnhardt	Jan	71
45 MX 100 Review	Bruce Detterich	Feb	45
Pascal From BASIC	David Sparks	Dec	67
Penguin Responds to Champagne Graphics		Nov	44
Response to Selection of a Disk System Part 2	George K. Peck	Oct	27
Selecting a Disk System	Dan Paymar	Jan	47
Selection of a Disk System Part 2	Dan Paymar	Oct	23
Spelling Programs	Ralph H Swerdlow, M.D.	Aug	31
The Inspector	Charlie Stillman	May	75
Two Disk Utilities	Val J. Golding	Jun	43
What's Where in the Apple	Val J. Golding	Jan	69
Word Processors	Jeffrey K. Finn	Jul	29
HUMOR			
BOOT	Craig MacIntosh	Jul	5
BOOT	Craig MacIntosh	Aug	5
BOOT	Craig MacIntosh	Sep	7
BOOT	Craig MacIntosh	Oct	7
BOOT	Craig MacIntosh	Nov	5
BOOT	Craig MacIntosh	Dec	7
So What Did You Expect?	Glen Bredon	Jan	16
So What Did You Expect Dept.?		May	54,70
So What Did You Expect Dept.?		Jun	7, 36
So What Did You Expect Dept.?		Jul	65
So What Did You Expect Dept.?	N.D. Greene	Aug	16
So What Did You Expect Dept.?	Frank Heffner	Oct	9
So What Did You Expect Dept.?		Nov	9
		32, 44, 69, 71	
So What did You Expect Dept.?		Dec	5
	37, 39 43, 46, 55, 59 66, 70, 71, 77, 80, 82		
MISCELLANEOUS			
A Call -A.P.P.L.E. Contest: So What Did You Expect Dept.?		Aug	67
A Call -A.P.P.L.E. Quickie		Sep	9, 39
A Call -A.P.P.L.E. Quickie: Trig With Integer Basic	Mark Cross	May	91
A Fourth Switch Input	Joel Buckley	Sep	51
An Open Letter to Apple Owners	Mark Pelczarski	Apr	57
A.P.P.L.E. Club Information: 10,000th Member Joins		Apr	7
A.P.P.L.E. Doctor	Henry Roberts	Jan	63
A.P.P.L.E. Doctor	Henry Roberts	Jun	73
A.P.P.L.E. Doctor	George Spelvin	Nov	83
A.P.P.L.E. Doctor	Cornelis Bongers	Dec	79
A.P.P.L.E. INFO/A Live Wire		Jun	85
A.P.P.L.E. Source Info		Oct	77
Business Software	Michael Thyng	Sep	69
Call -A.P.P.L.E. Contest Winners: Out of Sorts		Mar	51
1981 Call -A.P.P.L.E. Index	Kevin O. Donohoe and Val J. Golding	Feb	77
Character Generator ROM	Ian M. Jackson	Nov	21
Copy Protection and the Software Producer	David A. Lingwood	Apr	45
/// Corners Keypad Joystick	Richard V. Grant	Jul	41
Curing A Shiftless Apple	Dan Paymar	May	63
Disk Mapping the Z-80 CP/M System	Art Messler	Dec	45
Extra, Extra	Craig Peterson	Dec	55
Faire Enough	Val J. Golding	May	7
Keep Your Cool	Harry E. Brawley, Jr. and Art: Robert A. Kidder	May	57
NEXT	Wes Huntress	Dec	43
Open Letter to Computer Widows	Sabine Kurio	Jun	63
Our Biggest Contest Yet!		Dec	77
Our Smallest Contest Yet!		Dec	78
Piracy: Another View	Jerry Brieger	Apr	51
Protection: A Sordid Subject	Barry W. Collins	Apr	55
Robotic Series Overview and Orientation		Oct	69
Robotic Series	Michael Thyng	Dec	75
So What Did You Expect Dept.?		Jun	7, 36
So What Did You Expect Dept.?	Frank Heffner	Oct	9
So What Did You Expect Dept.?		Nov	9
		32, 44, 69, 71	
So What Did You Expect Dept.?		Dec	5
	37, 39, 43, 46, 55 59, 66, 70, 71, 77, 80, 82		
Staying Cool		Sep	48
Stack and Status Trace For Apple Forth	Guy T. Grotke	May	73
The First Annual Pacific Science Center & APPLE Festival	Michael Thyng	Sept	77
The Ghost in The Machine	Jonathan V. Post	Nov	72
The Protection Racket	Val J. Golding	Apr	43

TITLE	AUTHOR	ISSUE	PAGE
Tomorrow's Apples Today		Jun	71
Tomorrow's Apples Today		Jul	69
Tomorrow's Apples Today	Kevin Donohoe	Aug	69
Tomorrow's Apples Today	Kevin Donohoe	Sep	71
Tomorrow's Apples Today	Kevin Donohoe	Oct	78
Tomorrow's Apples Today	Kevin Donohoe	Nov	75
Tomorrow's Apples Today	Kevin Donohoe	Dec	83
Video Follow Up	Jim Hockenhull	Dec	37
VISI /// Utility	John Mancino	Jan	35
Write -A.P.P.L.E.	The Readers	Feb	74
Write -A.P.P.L.E.	The Readers	Mar	74
Write -A.P.P.L.E.	The Readers	Apr	72
Write -A.P.P.L.E.	The Readers	May	85
Write -A.P.P.L.E.	The Readers	Jun	79
Write -A.P.P.L.E.	The Readers	Jul	83
Write -A.P.P.L.E.	The Readers	Aug	78
Write -A.P.P.L.E.	The Readers	Sep	67
Write -A.P.P.L.E.	The Readers	Oct	73
Write -A.P.P.L.E.	The Readers	Nov	71
Write -A.P.P.L.E.	The Readers	Dec	82
Writing the Wrongs	The Wrongdoers	Apr	7

TUTORIALS

Adding Machine	Ralph Swerdlow, M.D.	May	15
Adventures With a RAMCARD	Charles Kluepfel	Sep	53
Basic Suggestions	Luis M. Hernandez	Dec	9
Business Software	Michael Thyng	Sep	69
Check Book Balancer	Ralph Swerdlow, M.D.	Jan	31
Color Me Apple	M.A. Iannce	Nov	9
Call -APPLE Contest Entry Comments	Luis A. Arana	Mar	57
"Disk Zap": Changing DOS Error Messages	Barton Feroe	Oct	29
Doing It With A Naked Apple Relocating Machine Language	Bill Starbuck	Dec	15
Envelope Addresser	Ralph Swerdlow, M.D.	Oct	9
Examining Directory Sector	David Huffman	Jul	15
Examining the Video Text Buffer	Bill Parker	Feb	9
Executive Text Files	Ralph H. Swerdlow	Jul	9
Graphic Garbage Collection	Richard Cornelius and Melvin Zandler	Nov	53
GETting Straight	Michael Thyng	Jun	67
Hexadecacolor Double Bubble Demo Sort	Scott Simpson	Mar	53
How About Arrays?	Ralph Swerdlow, M.D.	Apr	17
In aMIDSt Strings	Michael Thyng	Jul	67
Input Routine Handling	Michael Thyng	Feb	61
Ins and Outs: Part	Tom Little	Jun	25
Ins and Outs: Part 2	Tom Little	Jul	19
"Let's RESUME At Your Place"	Don Taylor	Jun	19
Making Mailing Labels	Ralph H. Swerdlow, M.D.	Jun	65
New Uses for the RAM Card	Mark Harris	Dec	25
Notes From A Novice	Charles Rusch	Jan	51
Notes From A Novice	Charles W. Rusch	Mar	65
Our Logical Apples	Ralph Swerdlow, M.D.	Feb	17
Pop Art	Ralph Swerdlow, M.D.	Mar	9
Printer Protocol	Ralph Swerdlow, M.D.	Sep	17
Programming and Problem Solving	Dr. David A. Lingwood	Dec	49
Program Optimization	David H. Bartley	May	9
Print ASCII	Ralph Swerdlow, M.D.	Dec	65
Random Adventures	David Sparks	May	37
Roasting Applesoft Preserves	David G. Sparks	May	47
Things About Strings	Michael Thyng	May	79
Total Amnesia	Val J. Golding	May	67
To Peaceably Assemble	David G. Sparks	Nov	57
Video Interfacing	Jim Hockenhull	Jun	9
Views on Hues	Michael Thyng	Oct	71

TECHNIQUE

A.P.P.L.E. Doctor	Henry Roberts	Jan	63
A.P.P.L.E. Doctor	Henry Roberts	Jun	73
A.P.P.L.E. Doctor	George Spelvin	Aug	75
A.P.P.L.E. Doctor	George Spelvin	Nov	83
A.P.P.L.E. Doctor	Cornelis Bongers	Dec	79
Applemash	Michael Thyng	Dec	39
Call -A.P.P.L.E. Contest Entry Comments	Luis A. Arana	Mar	57
"Disk Zap": Changing DOS Error Messages	Barton Feroe	Oct	29
Doing It With A Naked Apple Relocating Machine Language	Bill Starbuck	Dec	15
Doing the Splits	Roy Meyers	Aug	61
DOS/Pascal Bilingual Disks	Mat Davis	Oct	17
Dual Drive Master Create	David L. Deever	Jul	65
Dynamic Loading and Linking	John L. Shanks	Jul	59
Execute Strings	Cornelis Bongers	Sep	9
Executing Deferred Commands From the Keyboard	C. Dorsman	Aug	40
Expanding CHRGET	Anthony Rose	Sep	57
Fooling the Language Card	David Kelly	Feb	69
Garbagemen Strike Selective String Preservation	Clay Ruth	Aug	9
Get File Information Subroutine	Lou Infeld	Jan	19
Getting the High Sine: Parsing String Formulas	Charles Kluepfel	Mar	27
Hexadecacolor Double Bubble Demo Sort	Scott Simpson	Mar	53
Hi-Res Hi Jinks: Hi-Res Full Scroll	Edward C. So	Feb	23
Hi-Res Hi-Jinx	Edward C. So	Apr	59
How About Arrays?	Ralph Swerdlow, M.D.	Apr	17
IF/THEN/ELSE	Cornelis Bongers	Oct	57
Imbedding Machine Code In Applesoft Program Files	William J. Cheeseman	Jul	51
Improved Error Handling	David Short	Jun	22
"Let's RESUME At Your Place"	Don Taylor	Jun	19
Local Variables in Applesoft	Charles Kluepfel	Oct	13
Mathemagics	Michael Thyng	Mar	33
Mod Art	Wayne Eastwood	Mar	37
New Uses for the RAM Card	Mark Harris	Dec	25
Our Logical Apples	Ralph Swerdlow, M.D.	Feb	17
Passing Array Names	Robert Purves	Oct	46
Patching DOS the Easy Way	Leighton Paul	Jul	49
Pitch & Rhythm on the Apple	Jack H. Bender	Jun	15
Pointers on Pointers	John Rodger	Jan	75
Pop Art	Ralph Swerdlow, M.D.	Mar	9
Print Using Finds a Friend	Craig Peterson	Oct	55
Programming and Problem Solving	Dr. David A. Lingwood	Dec	49
Random Ideas	Val J. Golding	Apr	63
Slow Plot	Jim Morriset	Nov	63
The Unknown INTFLG	Kelly W. Puckett	Dec	23
Trig or Treat	Raymond Usher	Mar	31
Using a Frimware Card in Slot 4	David Morganstein	May	69
Vandal Foiler	Leighton Paul	Feb	71
Vandal Foilers Foiled, Editor Omits Program		Mar	76
Weed Out the Trash	Clay Ruth	Nov	68
Writing Text Files From the Keyboard	Charlie Stillman	Apr	52
Yes, You Really Can Put Files on Track 0	Clay Ruth	Dec	57
Young Blood: Preventing Accidental Deletions	John Stahl	Feb	73
Z-80 Card Fix	P.C. Lincoln	Apr	69

UTILITIES

A New DOS Command: Filedump	Lee Reynolds	Jun	47
Applesoft Preserves	Val J. Golding	Feb	65
Binary To Text Converter	Val J. Golding with Dave Lingwood	Jul	77
Converting Apple Writer Files	Frank D. Kirschner	Feb	57
Converting Applewriter Files	Thomas G. Williams	Dec	31
Ctrl-C Killer	S. Scott Zimmerman	Dec	71
Disk Verify	Roger Chaffee	Jun	39
Dual Boot	Bill Bush	Jun	49
Find It	Anthony Rose	Oct	63
Free Space on the Screen	Mike Firth	Sep	47
Free Space Patch	Val J. Golding and Mark Pump	Apr	65
Get Most Followup	Clay Ruth	Apr	71
Get the Most From a Diskette	Clay Ruth	Mar	13
Hi-Res Full Scroll	Edward C. So	Feb	23
In the Dumps	Val J. Golding	Oct	37
LCASE KILLER	Eric Lambrecht	Mar	70
Moving Big Mac Symbol Table	Thomas G. Williams	Jul	79
Not Illegal Direct	Lee Reynolds	Jan	33
Pager +	Gary Little	Oct	49
"Patching" the "Free Space"	C.J. Welman	Jun	59
Print to Disk	Vince Giambalvo	Aug	51
Random Ideas	Val J. Golding	Apr	63
Remsaver	Eric Lambrecht	Sep	59
Restore	Eric Lambrecht	Apr	61
Restore To A Line Number Routine	Lee Reynolds	Jul	43
Returning Your Remarks	David Sparks	Oct	67
Roasting Applesoft Preserves	David G. Sparks	Aug	47
Text File Fast Loader	Robert Purves	Sep	39
To Peaceably Assemble	David G. Sparks	Nov	57
Using MAC.LC With Firmware Card	David Deever	Sep	63
Vandal Foiler	Leighton Paul	Feb	71
Vandal Foilers Foiled, Editor Omits Program		Mar	76

StarLogic Announces Apple II Compatible Disk Drives...$225

Includes drive, cable, cabinet and 90-day warranty
Also compatible with Franklin ACE

TELEPHONE ORDERS ONLY
MASTERCARD OR VISA REQUIRED
(213) 883-0587

StarLogic

Apple II is a registered trademark of Apple Computer, Inc.

Help Wanted?

ASSISTANT PROGRAMMER™
Menu Generator and Screen Text Editor

- Accepts all characters from keyboard (including " [/ __ ").
- Shifts individual lines or entire page with I-J-K-M keys.
- Splits a line or combines two lines at cursor position.
- Inserts input or variable print commands at cursor position.
- Spaces added or deleted between lines with a single keystroke.
- Text file created to "EXEC" menu (BASIC) into your program.
- Accents your titles with borders, using minimum amount of bytes.
- No disk swapping, entire program resides in memory for easy access.
- Text file created that reinstalls screens for future development.
- Prints copy of menu in screen format with row and column edge scales.
- Renders line at cursor position inverse or flashing with keystroke.
- Overhaul your existing menu with transparent capture routine.
- Gives printed list of input/print commands with row & column numbers.
- Right, left and center justification of menu at cursor position.
- All features and procedures fully documented and tutored.
- Menus can be designed on the screen, no need for graph paper layouts.
- Makes horizontal lines print in vertical format at the touch of a key.
- Error messages displayed for all options (display voidable).
- Return coupon for your ticket to fast, efficient and friendly menus.

Requirements: Apple II Plus* with 48K & disc drive.
*Apple II is reg.trademark of Apple Computer, Inc.

ALL FOR ONLY $49.95 To order send check or money order. Pa. residents add 6% tax. COD add $3.00 extra.

Soph-Key
Box 387
Flourtown PA 19031
(215) 836-1056

SELL YOUR SOFTWARE

ADVERTISE IN THE LAMA SOFTWARE DIRECTORY

The LAMA Software Directory is a high quality inexpensive way for both large software companies and single APPLE users to advertise their application and system software for the APPLE Computer. A listing in the LAMA Software Directory includes:

Program Name
Program Description
APPLE Configuration
Prices and
Ordering Information

and costs just $15.00 for ONE FULL YEAR. In addition, orders of two listings or more receive a 33% discount, lowering your cost to just $10.00 per listing.

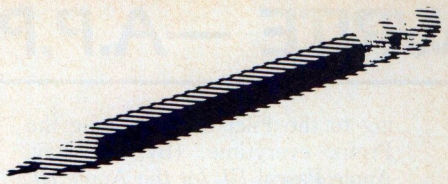
The LAMA Software Directory will be available at all participating APPLE dealers and bookstores beginning in January, 1983. In order to be included in the first issue we must receive your order by November 30, 1982. New issues will follow every four to six months.

For more information send your name and address to:

The LAMA Software Directory
P.O. Box 201
Bloomingdale, IL 60108
(312) 830-4159

APPLE is a trademark of Apple Computer, Inc.

WRITE -A.P.P.L.E.



The Readers

Quadriplegic Help

Dear Val,

DURING my vacation in April 1982, I met a young man named Mike McIntyre who is quadriplegic. Mike has an interest in learning computer programming with an ultimate goal of self support. Since I have had 25 years background in Electronics with 15 associated with computers I wanted to help Mike if possible. He is able to use his mouth to draw some very nice pictures and seems mentally sharp.

I have subscribed to some magazines and enrolled him as a member of A.P.P.L.E. I would like to do more but I an not sure what direction I should take. Hopefully the readers of *Call - A.P.P.L.E.* could give me some direction. I believe some friends of mine will assist in purchasing a computer for Mike if he continues to show interest. Possibly there is some adaption to the Apple to assist the handicapped.

Any assistance the readers can provide will be appreciated. Mike's address is as follows:

Mike McIntyre
Worthington Manor Cane Center
Cone Road and 36th Street
P.O. Box 4276
Parkersburg, W.VA 26104

Thank you in advance

H.K. Moreland
c/oAramco Bx 8109
Dhahran, Saudi Arabia

Patching The Lister

Dear Val,

I am grateful to Wayne Eastwood and A.P.P.L.E. for The Lister in the A.P.P.L.E. Applesoft Carpenter. I offer the following patches in hopes other users will find them useful.

I wanted to have the default setting for WIDTH to be 80 columns and not 38, since I make formatted listings only on my printer, which is 80 columns wide. To make the default WIDTH 80 columns, change the $26 at $9045 to $50, and then change the $B3 and $B8 at $8C52 to $B8 and $B0 respectively.

To use less paper, I wanted to eliminate the default skip-a-line between succeeding lines. To do this I changed the $01 at $9046 to $00, and also the $B1 at $8C79 to $B0.

A couple of other notes: If one has a real-time clock and a time/date output subroutine, one can replace the JSR at $8A5B with a JSR to one's time/date output routine. This will print the time & date on the line after the "FILE".

Also, since I have the ApL.Case lower case chip, I installed the 4-byte conversion patch on page 12 of the manual. But when I enter "&H" the "help" screen comes up with gibberish. Haven't straightened this one out yet.

Again, many thanks for the great software and publications. Keep up the good work.

Sincerely
Stephen Bach
Rt. 2 Box 89
Scottsville, VA 24590

Software Solution to $CFXX

Dear Val,

In David Goss's article "Beware the $CFXX," Mr. Goss only covered the hardware solution to the problem of Applesoft turning off the expansion ROM if $CFXX decoding is used in the design of a peripheral card.

A software solution also exists to the problem. All that is required to impliment it is a little forethought in the writing of the program which is to be burned into ROM.

Mr. Goss fails to mention it, but I assume that he is directly accessing the routines he had stored on his ROM. Had he jumped to those routines via the slot PROM he would have never noticed the Applesoft problem since the $CnXX (n = slot number) access would have re-enabled his ROM.

Solution, only call your expansion ROM routine via the slot PROM as such:

```
$XXXX JSR PROM
$CnXX JMP EXP. ROM
$C800 EXP ROM ROUTINE
```

Also an access of $CFFF in the slot PROM routine prior to the jump to the expansion ROM would ensure that all expansion were turned off.

```
$XXXX JSR PROM
$CnXX STA $CFFF
$CnXX JMP EXP. ROM
$C800 EXP ROM ROUTINE
```

The simple use of this technique solves all expansion ROM contentions and allows one to use the $CFXX decode without problems. It also helps save power by cutting down on the number of integrated circuits required to make the card work, and considering the Apple][power supply every bit helps.

Another helpful hint to hardware hackers.

Got a peripheral card routine that will only fit in the slot PROM if written in slot dependent code? DO IT!!! Eight 256 byte passes can fit in a 2716 EPROM (but only seven are required since the Apple has no slot 0 PROM). With the A0-A10 lines and the D0-D7 lines connected to their counterparts on the card edge connector, connecting the CS and OE lines to the I/O STROBE completes the circuit giving you seven slot dependent slot PROMs on one chip with zero support chips.

Also connecting pin 21 of the EPROM to the R/W line the edge connector allows one to replace the 2716 with a 2016 or similar 2048*8 bit static RAM for easy program development and debugging.

Matthew Stier
USS NYC SSN 696
FPO SF, CA 96673

Poking Pascal Prefix

Dear Val:

My copy of "All About Pascal" arrived last week and it was great fun reading through it. I found the Memory Map for Pascal 1.1 especially helpful. It provided the answer to two problems I had been thinking about for some time. Since other members may find the answers useful, I pass them along herewith.

1. Because the Apple disks have a fairly small capacity, with all the files that have to be around when Pascal is running, I tend to use disk drive 2 (unit 5, in Pascal terms) as the working volume, and to swap disks in and out of it frequently, keeping the basic system files (Editor, Filer, System.Library, etc.) in drive 1. Therefore, I routinely set the Prefix to "#5" on starting up for the day. With Pascal 1.1, however, the Filer will only set the Prefix to a volume name, not a device—so "P#5" only reads the directory of the disk that is in drive 2 at the time and sets the Prefix to that disk's name. I prefer to reference only the drive, so I can move disks in and out of drive 2 as needed without go-

—●→

ing to the Filer and changing the Prefix everytime. (By the way, Apple Pascal *III*, for the Apple *III*, does permit setting the Prefix to a device (drive), as well as a volume. That's where I got the idea.)

Solution

POKE "#5" into memory at the location where the interpreter expects the Prefix name to be. Thanks to the memory map at pages 75–78 of "All About Pascal," we now know that that location is an eight-byte area beginning at $AA08 (or -22008 in decimal).

So, I wrote a SYSTEM.STARTUP program to do the appropriate POKEing. The program following my letter is reduced to its basic scheme. Using the "variant record" technique — declaring a record that defines a two-byte quantity as either an integer or a pointer to the address in memory containing a variable — I was able to POKE the string "#5" in one step, rather than a character at a time.

2. Many of the programs that I write for use by my law firm are quite long, and use fairly long library UNITs. The result is that I need to use both the Compiler swapping options and the system swapping option to avoid stack overflow errors during compilations. Since I consistently forget to toggle the system swapping option, it would be very handy to have that done automatically when the system is booted. For a while, I used a SYSTEM.STARTUP that chained to an Exec file which executed the appropriate commands. Now I know that the system swapping flag is located at $AC3E (-21442 in decimal). A little experimentation established that a 0 at that location signifies swapping "off," and a -1 (that is, $FFFF) signifies swapping "on." (Probably, any non-zero value would do, but -1 works fine.) So, POKEing -1 into memory at -21442 in the SYSTEM.STARTUP program turns on system swapping quickly and neatly.

The listing also shows a POKE to set up my Sup'R'Term 80-column board to initialize it for lower-case and "true" shift key action. It may be that other cards can be initialized in a similar way; but it would require analysis of the firmware for a particular card to see how it uses flags and where it stores them.

The SYSTEM.STARTUP that I use regularly also reads the current date and time from my Thunderclock card and updates the system date. I have not included that part of the program in the listing, because the Geddes/DeGroat procedure in "All About Pascal" shows how to set the system date, and reading the Thunderclock involves external assembly-language routines that would take us beyond the intended scope of this simple program.

David W. Walker
16 Rockland Street
Wellesley Hills, MA 92181

```
program STARTUP;

{ SYSTEM.STARTUP program to:
        set Prefix to '5';
        set Swapping toggle 'on';
        set Sup'R'Term 80 column keyboard to
          --lower case input
          --hardware shift key enabled; }

{ Could also include a routine to set
  the system date,
  by reading a clock card or by
  calling a procedure
  such as Geddes and DeGroat's
  "GETDATE",
  Call -A.P.P.L.E. in Depth -- All
  About Pascal, Page 117 }


type    volid  = string[7];
        bytef  = packed array[0..1] of 0..255;
        magic  = record case integer of
                   1 : (Addr : integer);
                   2 : (Str : ^volid);
                   3 : (Flag : ^integer);
                   4 : (Byte : ^bytef);
                 end;

var Id : volid;
    Sys : magic;

begin

  gotoxy (3,8);
  writeln ('+++Welcome to Apple Pascal 1.1 +++');
  writeln;
  writeln ('    The Swapping flag is ON.');
  writeln ('    The Prefix is ''#5''');

  Id := '#5';
  Sys.Addr := -22008 { $AA08 address of prefix in memory }
  Sys.Str^ := Id;    { set prefix to '#5'                 }

  Sys.Addr := -21442 { $AC3E -- system swapping flag }
  Sys.Flag^ := -1;   { swapping on                   }

  Sys.Addr := 1403     { $57B -- Sup'R'Term flag byte }
  Sys.Byte^[0] := 64; { set bits 2, 5, and 6         }

end
```

REL Bug

Dear Editor

For those of us Apple][hackers that have been using the original "Apple 6502 Assembler/Editor" in the DOS 3.3 Tool Kit, I am sure you have probably been confused about the REL pseudo-op, and the funny "R-type" binary file it generates. Well, I certainly was confused and I spent a weekend playing with it.

I have several machine language programs (a print using routine, a Hi-Res picture compression routine, RWTS driver, a SILENTYPE driver/spooler that uses the language card, etc.) and I have had to use them all in page 3. After discovering what the relocating feature of the ASSEM/ED did, I was no longer satisfied with page 3! However, as they say "There is always one more bug," and I am happy to say I found one in the RBOOT program that comes with the Tool Kit. On page 57 of "Apple 6502 Assembler/Editor" book it describes the BASIC statements needed to activate RBOOT and RLOAD. In line 30 of the

BASIC program the file name of the machine language routine follows: 30 ADRS=USR(0). If the file name enclosed in quotes has a space in the name, such as: "PROG ONE," the error message "UNDEF'D STATEMENT ERROR IN 0" will be generated. So it is necessary that all machine language routines using the REL feature must be one word filenames.

Thank you for such a great magazine as *Call -A.P.P.L.E.*, and I hope that I can be of future help.

Thomas J. D. Straub
4310 Brookhead Trail
Houston, TX 77066

Peachy Problems

Dear Val,

Five months ago, I bought three Peachtree Software programs from *O.M.C.* of 20695 S. Western Ave. #124, Torrance CA. namely A/P 40 ser: 1837, G/L 40 ser:2568 and A/R 40 ser:3041).

In general terms, the programs have done what I expected them to do for me, but now that I use the G/L regularly, for instance: I input some 100-140 transactions per month, and have seven departments (in my case projects). To run the trial balance takes more than 25 minutes with only three minutes actual printing time (the time the ready light from my epson printer is off); then I get my balance sheet (no problem there); and finally I get my income statement which again takes more than twelve minutes, and since I need departmental income statements, these take another seven times eleven minutes to finish.

My basic questions are the following:

1. Would a Disk-Emulator help me, if so which one should I get and just how much would it improve?
   My system comprises: Apple II plus Epson MX80 FT + Indigo 16K RAM/Language + Microsoft Softcard and a couple of Apple Disk drives.
2. In order to process and print-out several department income statements one after the other, is there a way to avoid having to go back to the main menu, reload the selection every time, and go through all the questions, date, etc?
3. If during November, I discover a mistake in September, I understand I should make a correction transaction for November, leaving therefore: September, October and November with inaccurate information. Or is there a way to modify September, process it and get it's results correct; then without having to redo October just modify its starting data and process it to also be correct with which I believe everything stays just fine.

I thank you in advance and hope to hear from you soon.

Eduardo Mariscal
App.451, Col.del Valle
Nuevo Leon, Mexico 66220

A DISK-EMULATOR would probably help by 25-30% improvement, assuming *that Peachtree software is not protected. Not being familiar with the Peachtree programs we cannot answer the other questions, however perhaps one of our readers can.*

So Wrong, Jim

Dear Val,

I am flattered that you chose to publish my "So What Did You Expect?" in the November issue (p.69). There is a small omission, however, in the listing as published. The GR was inadvertently left out of line 10, which should read:

```
10 TEXT : HOME : GR
```

As you might imagine, the program isn't too exiting when run without the graphics mode set.

Jim Ganz
82 Spring Lane
West Hartford, CT 06107

Applewriter Escapes

Dear Val,

Thanks for your letter of 10/6/82 concerning Applewriter][and the embedding of ESC control characters.

I have been away and just got to your note. I appreciate your follow-up.

I have solved my problem which might have been obvious to you and Applewriter's authors but is not to a novice. In case someone else has this problem:

"Ctrl-V" "ESC" "ESC" "P" "Ctrl-V"

is what is required. I was trying: Ctrl-V ESC P Ctrl-V and it didn't work. The extra ESC is needed to signal a capital P. I thought Ctrl-V took you temporarily out of Applewriter and that the "P" after ESC would be seen as a capital.

Gregory J. Lazarchik
210 Seneca Drive
Mt. Lebanon, PA 15228

Bugs in the Garbage

Dear Val,

While relaxing after Thanksgiving dinner (actually, I was too stuffed to move), I began working on a general purpose routine to convert hex to decimal and back. Of course, out came in Depth #1 and a couple of conversion routines therein. Page 14 of particular interest. It seems that there is an error that does not allow the sequence of hex characters "DEF" to be handled properly for hex to decimal conversion.

Every conversion attempt from hex (&%xxxx) to decimal with the sequence "DEF" fails. &3567 generates 0DEF as output. I am not sure what to think of this.

On another subject, Richard Cornelius and Melvin Zandler are to be congratulated on their fine presentation with *Graphic Garbage Collection.* I can certainly appreciate the amount of effort that went into building this very simple demonstration of Applesoft string handling.

Visually, I could immediately see the benefit of Randy Wigginton's fast garbage collection routine. However, certain obstacles had to be overcome first. For one, the hex dump of Fast Garbage presented on page 55 does not match Randy's original work from January 1981. It is 12 bytes longer. The 12 bytes are the sequence beginning at $9359. This code apparently clears a zero page location and saves the pointer for top of array storage at some unused location. Also, the value for NUMELS is set to 32 compared to 16 in the original version.

The end of the article states that a CALL 37700 will enable AUTOMATIC garbage collection. Like the original article, it fails to state that the user must place the CALL in his BASIC program to get the fast garbage collection. It is not automatic.

Randy's routine can be made automatic by using Jim Morriset's routine on page 64 to move Applesoft to RAM and placing JMP $9344 at location $E384.

While performing this little feat of magic, I encountered one problem. Jim's routine will not work unless the HELLO routine on the DOS 3.3 System Master loads Integer first. Normally, I do not load Integer into the RAM card. Jim's routine would lock up my Apple attempting to access RAM (even RESET would not work). Once Applesoft is moved, the RAM card is opened for write access, and $E384 is changed to use Randy's fast garbage collection routine.

With this complete, I timed the graphic demo with Applesoft garbage collection. The elapsed time was seven minute and 40 seconds. Then, I timed Randy's routine. Unfortunately, Randy's routine is set to garbage collect when free space drops below 1K. The graphics demo invoked fast garbage collection for each of the last 120 elements of the G$ array. This caused the elapsed time of the graphics demo using fast garbage collection to be over 10 minutes (I stopped timing at 10 minutes). By changing the free space threshold to 20 bytes or using the modified Applesoft in RAM, the time dropped to a more respectable one minute and 54 seconds.

Anyone using Randy's routine should drop the threshold to a small value. Otherwise, if free string storage falls below the threshold and stays there, garbage collection is invoked almost continuously.

Be aware that Randy's routine does not set HIMEM. This must be done to protect it from being overlaid by strings.

Ken Kashmarek
6 Cherokee Ct. RR2
Eldridge, IA 52748

YOU MAY HAVE SOLVED one of the problems we had with Garbage. Thanks. You also stumbled on an obscure bug in our Hex/Dec Converter program. What actually happens is that when the user enters a hex value starting with DEF. . ., the Applesoft interpreter parses in the $B8 token used for DEF FN and it thus goes unrecognized by the converter. We suspect it would take several more lines of code to trap the B8 and force the correct value in.

Dear Val,

Here is some evidence as to how important your articles are — at least one of them. The article I refer to is *Garbagemen Strike* by Clay Ruth in the August 1982 issue.

Wow!!! What a help!!!!

About a year or so ago, I took a microprocessor training course and acquired an Intel SDK-85 kit which uses the 8080 machine language. Many hours have been expended on entering machine language code by hand with no easy way to save or store programs after power was removed.

Then several months ago, I met a friend at work that was part of a group who were writing a program entitled "Apple][to Intel SDK-85 Teletype Interface." This program, which was ultimately purchased from Datco, P.O. Box 72, Windsor, NY 13865, was able to use the Apple and Disk System to Display Memory (D), Execute Programs (G), Insert Instructions (I), Move Memory (M), Substitute Memory (S), Exam/Modify 8085 CPU Registers (X) similar to those available for the Apple's Monitor System. In addition, with the Apple's Disk System, one can display the disk catalog; Load programs, from disk to Apple memory; Save programs from Apple memory to the disk; Load program from Intel memory to Apple memory; load programs from Apple memory to Intel memory; Load and Save programs input by keyboard; Review/Modify data or programs in Apple memory.

Trying to follow and understand programs displayed in the hex machine code is difficult, so I have added an 8085 disassembly program which I wrote myself.

This is where my problem started. I have a large program approx. 1500 bytes, that operates on the SDK-85 kit. A disassembly listing of this program took approx. 30 hours to print, due mainly to the small amount of memory left on the Apple rapidly filling up with the string variables. Six lines of output were printed at a time, then the Apple had to clear the garbage.

I then received the *Call -A.P.P.L.E.* issue, with the article that solved the problem. With judicious use of the two subroutines mentioned, the printing time was reduced to 45 minutes. So you see why I would whole heartedly recommend your information exchange.

Keep up the good work.

John W. Butler
57 Elmwood Drive
Apalachin, NY 13732

Doing the Shuffle

Dear Val,

Thanks to Vincent Aderente for his up-date on a playing card generator in November 1982 *Call -A.P.P.L.E.* I recall a few years ago you published some shuffle routines for playing cards. As I recall they were a bit cumbersome using arrays and flags.

Several years ago, I wrote the enclosed program in Applesoft to randomly shuffle a deck of cards. The concept is to randomly remove one card element at a time from a string which represents an unshuffled deck and concatenate that element to a new string which would represent the shuffled deck. Removal is accomplished by separating the left elements to a new string which would represent the shuffled deck. Removal is accomplished by separating the left elements and right elements from the selected element and then concatenating the resulting left and right elements together into a new string which now no longer contains the selected element. Thus the same card cannot be chosen more than once. The concept is so simple that I can't be sure that someone hasn't thought of it or has previously published it.

In the program, N equals the number of characters in each element. In line 10, A$ is the unshuffled card string, 0 represents decimal 10 and J,Q,K,A are the picture cards with H,D,C,S being the suits.

Line 20 checks to see if the length of the unshuffled deck string is down to the last two characters, N and if so they are concatenated to the end of the shuffled deck NB$. In line 22, S represents the randomly selected element (1 through 52) from the unshuffled deck string and C defines the location of the first character of the element. S$ then becomes the string of that selected element.

Lines 40 and 50 separate the selected element from the unshuffled deck string, if it is the first element or last element respectively, concatenating the selected element to the new shuffled deck string.

Finally, in line 60 an "interior" element is separated and then the remaining left ad right elements are concatenated into a new unshuffled deck string.

The shuffling takes less than three seconds. Cards could also be identified by two character numeric string format such as "01" to "52" and VAL command could be used to convert them later to numeric integer values. By changing the value of N, elements of other than two characters/element can be shuffled with string length limited by the maximum 255 characters.

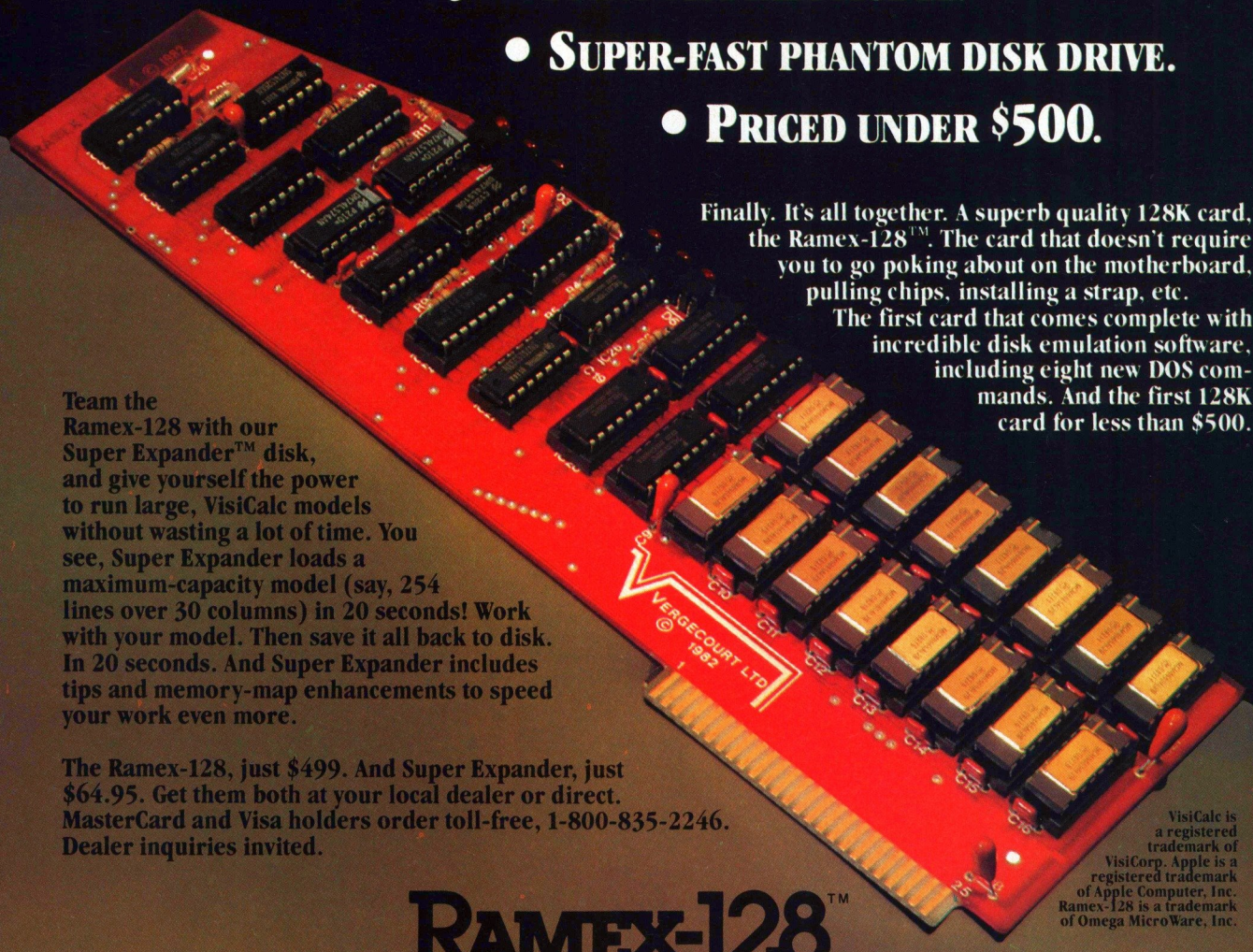
I hope this routine proves helpful for all *Call -A.P.P.L.E.* card sharks!

Maurice Bernstein, M.D.
8121 Van Nuys Blvd.
Panorama City, CA 91402

```
1   HOME :N = 2
10  A$ = "2H3H4H5H6H7H8H9H0HJ
       HQHKHAH2D3D4D5D6D7D8D9
       D0DJDQDKDAD2C3C4C5C6C7
       C8C9C0CJCQCKCAC2S3S4S5
       S6S7S8S9S0SJSQSKSAS"
20    IF  LEN (A$) = N
       THEN NB$ = NB$
       + A$: PRINT NB$:
       END
22  S =  INT ( RND (1)
       * ( LEN (A$) /
       N) + 1):C = (S
       * N) - (N - 1):S$
       =  MID$ (A$,C,N)
40    IF C = 1 THEN A$
       =  RIGHT$ (A$,
       LEN (A$) - N):NB$
       = NB$ + S$: GOTO
       20
50    IF C =  LEN (A$)
       - (N - 1) THEN
       A$ =  LEFT$ (A$,
       LEN (A$) - N):NB$
       = NB$ + S$: GOTO
       20
60  A$ =  LEFT$ (A$,C
       - 1) +  RIGHT$
       (A$, LEN (A$)
       - C - (N - 1)):NB$
       = NB$ + S$: GOTO
       20
70    REM

DECK OF CARDS SHUFFLE
       ROUTINE BY MAURICE
       BERNSTEIN, M.D.
```

NOW HERE FOR YOUR APPLE®: A CARD THAT BLOWS ALL THE OTHERS AWAY!

- LOADS OR SAVES A FULL VISICALC® MODEL IN 20 SECONDS. (NOT 16 MINUTES LIKE THE OTHERS)
- 136K FOR A VISICALC MODEL.
- SUPER-FAST PHANTOM DISK DRIVE.
- PRICED UNDER $500.



Finally. It's all together. A superb quality 128K card, the Ramex-128™. The card that doesn't require you to go poking about on the motherboard, pulling chips, installing a strap, etc.

The first card that comes complete with incredible disk emulation software, including eight new DOS commands. And the first 128K card for less than $500.

Team the Ramex-128 with our Super Expander™ disk, and give yourself the power to run large, VisiCalc models without wasting a lot of time. You see, Super Expander loads a maximum-capacity model (say, 254 lines over 30 columns) in 20 seconds! Work with your model. Then save it all back to disk. In 20 seconds. And Super Expander includes tips and memory-map enhancements to speed your work even more.

The Ramex-128, just $499. And Super Expander, just $64.95. Get them both at your local dealer or direct. MasterCard and Visa holders order toll-free, 1-800-835-2246. Dealer inquiries invited.

VisiCalc is a registered trademark of VisiCorp. Apple is a registered trademark of Apple Computer, Inc. Ramex-128 is a trademark of Omega MicroWare, Inc.

RAMEX-128™

UNDER $500. WHAT ARE YOU WAITING FOR?

OMEGA MICROWARE, INC.™

222 SO. RIVERSIDE PLAZA • CHICAGO, IL 60606 • 312-648-4844

Inside Apple

Apple Computer Inc., 20525 Mariani Avenue, Cupertino, California 95014 Vol. 1 No. 2

For the authorized Apple dealer nearest you, call **800-538-9696** (**800-662-9238** in California.)

Fruitful Connections.

There are more people in more places making more accessories and peripherals for Apples than for any other personal computer in the world.

Thanks to those people — in hundreds of independent companies — you can make the humblest 1978 Apple II turn tricks that are still on IBM's Wish List for 1984.

But now we're coming out with our very own line of peripherals and accessories for Apple® Personal Computers.

For two very good reasons.

First, compatibility. We've created a totally kluge-free family of products designed to take full advantage of all the advantages built into every Apple.

Second, service and support.

Now the same kindly dealer who keeps your Apple PC in the pink can do the same competent job for your Apple hard-disk and your Apple daisywheel printer.

So if you're looking to expand the capabilities of your Apple II or III, remember:

Now you can add Apples to Apples.

A joy to behold.

The new Apple Joystick II is the ultimate hand control device for the Apple II.

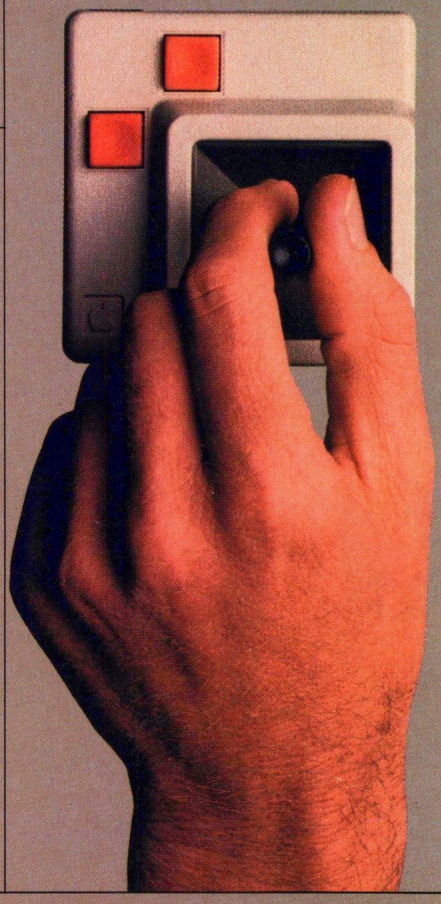
Why is it such a joy to use?

With two firing buttons, it's the first ambidextrous joystick — just as comfortable for lefties as righties.

Of course, it gives you 360° cursor control (not just 8-way like some game-oriented devices) and full X/Y coordinate control.

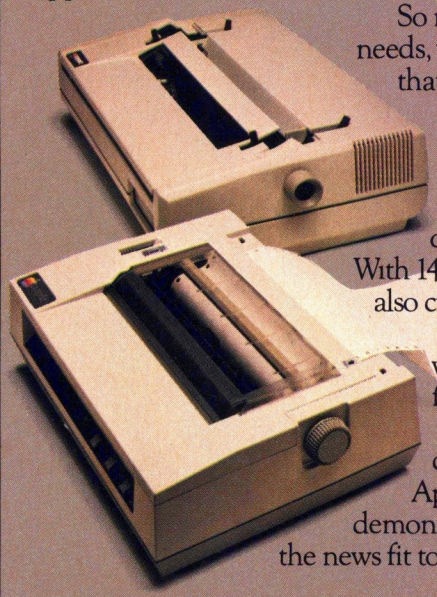
And the Joystick II contains high-quality components and switches tested to over 1,000,000 life cycles.

Which makes it a thing of beauty. And a joystick forever.



Gutenberg would be proud.

Old Faithful Silentype® has now been joined by New Faithfuls, the Apple Dot Matrix Printer and the Apple Letter Quality Printer.

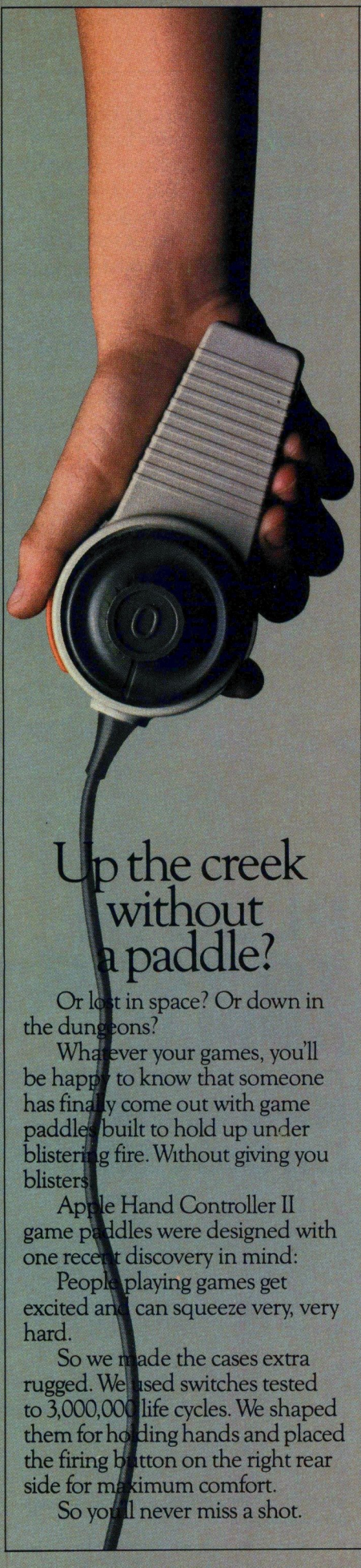


So now, whatever your budget and your needs, you can hook your Apple to a printer that's specifically designed to take advantage of all the features built into your Apple. With no compromises.

The 7x9 Apple Dot Matrix Printer is redefining "correspondence quality" with exceptional legibility. With 144x160 dots per square inch, it can also create high resolution graphics.

The Apple Letter Quality Printer, which gets the words out about 33% faster than other daisywheel printers in its price range, also offers graphics capabilities. See your authorized Apple dealer for more information and demonstrations. Because, unfortunately, all the news fit to print simply doesn't fit.

©1983 Apple Computer Inc.



Up the creek without a paddle?

Or lost in space? Or down in the dungeons?

Whatever your games, you'll be happy to know that someone has finally come out with game paddles built to hold up under blistering fire. Without giving you blisters.

Apple Hand Controller II game paddles were designed with one recent discovery in mind:

People playing games get excited and can squeeze very, very hard.

So we made the cases extra rugged. We used switches tested to 3,000,000 life cycles. We shaped them for holding hands and placed the firing button on the right rear side for maximum comfort.

So you'll never miss a shot.

A storehouse of knowledge.

If you work with so much data or so many programs that you find yourself shuffling diskettes constantly, you should take a look at Apple's ProFile,™ the personal mass storage system for the Apple III Personal Computer.

This Winchester-based 5-megabyte hard disk can handle as much data as 35 floppies. Even more important for some, it can access that data about 10-times faster than a standard floppy drive.

So now your Apple III can handle jobs once reserved for computers costing thousands more.

As for quality and reliability, you need only store one word of wisdom:

Apple.

Launching pad for numeric data.

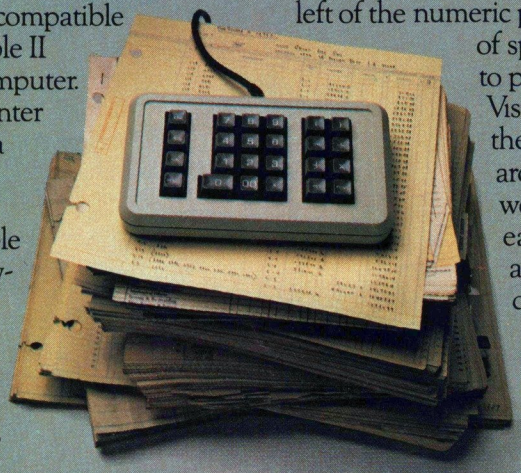
Good tidings for crunchers of numerous numbers:

Apple now offers a numeric keypad that's electronically and aesthetically compatible with the Apple II Personal Computer. So you can enter numeric data faster than ever before.

The Apple Numeric Keypad II has a standard calculator-style layout. Appropriate, because unlike some other keypads, it can actually function as a calculator.

The four function keys to the left of the numeric pad should be of special interest to people who use VisiCalc.® Because they let you zip around your work sheet more easily than ever, adding and deleting entries.

With one hand tied behind your back.



VisiCalc is a registered trademark of VisiCorp. Inc.

Plug yourself into a new world of possibilities for you and your APPLE.

Have you ever wished that your APPLE computer could do just one more thing?

To somehow perform that one task that would just exactly fit your particular need.

You may have found that a hardware limitation prevents you from accomplishing your goal and that there are no interfaces or expansion modules designed for your particular application. Are you frustrated? Not any more!

THE CUSTOM APPLE & OTHER MYSTERIES, volume one in IJG'S APPLE Information Series, provides you with the information, specifications and references you need to do it yourself, whether a novice or expert, and includes the basic information required for hardware enhancements that are common to many projects.

Guide to APPLE Hardware and software modification.

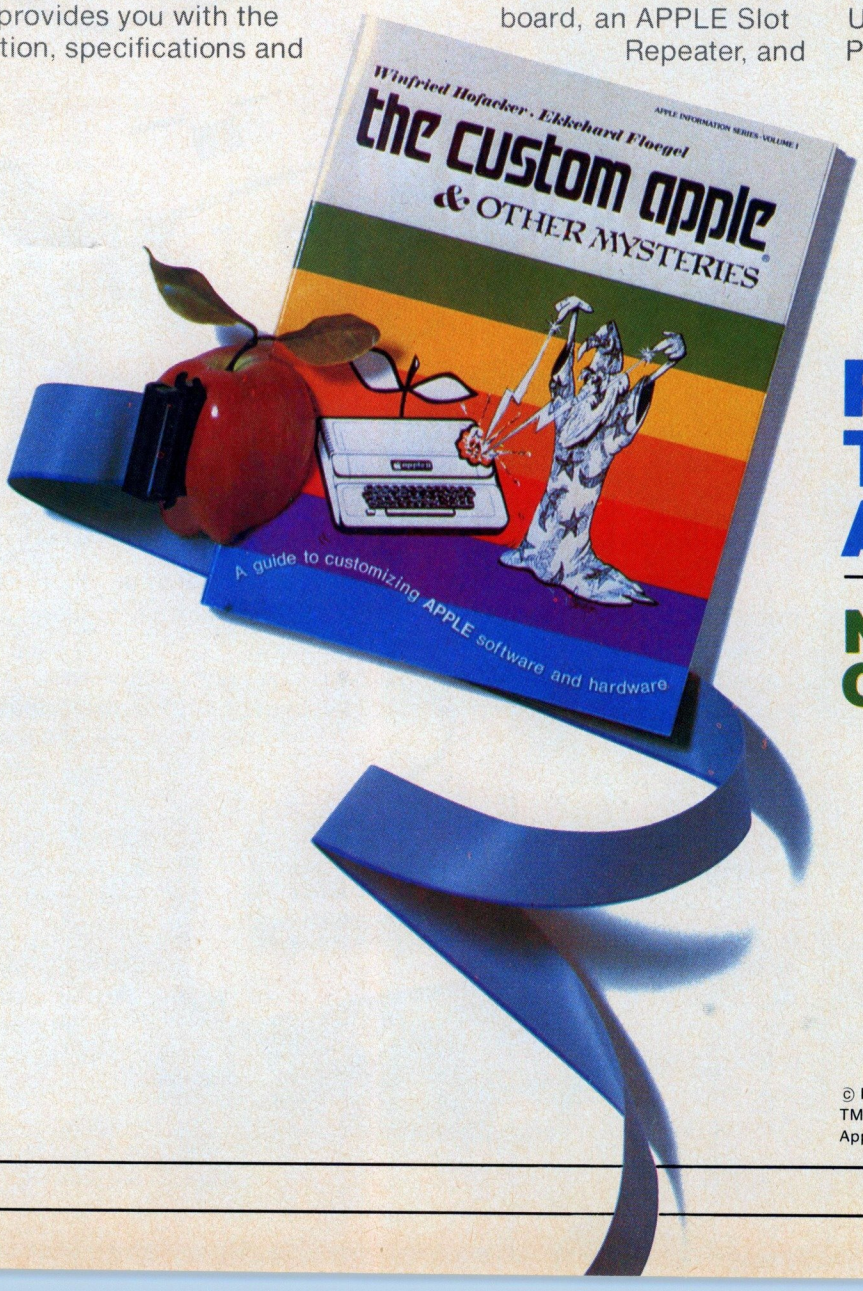
THE CUSTOM APPLE & OTHER MYSTERIES includes a number of data acquisition and control projects with printed circuit layouts like an 8-Bit D/A and A/D Converter, a 6522 application interface board, a sound and noise generator board, an EPROM Burner board, an APPLE Slot Repeater, and includes information on the APPLE as a square wave generator, the control of stepper motors, connecting two 6502 systems, and lots more.

The Custom APPLE & Other Mysteries is a valuable tool for all APPLE or APPLE II computer users and is available for $24.95 at computer stores, B. Dalton Booksellers and independent book dealers. If your dealer is out of stock, order direct from IJG.

Include $4.00 for shipping and handling. Foreign residents add $11.00 plus purchase price. U.S. funds only please.

1953 West 11th Street
Upland, California 91786
Phone: 714/946-5805

Helping You Help Yourself.



PLUG IN TO YOUR APPLE.

New Book On Sale NOW!

© IJG, Inc. 1982
TM APPLE and APPLE II are registered trademarks of Apple Computer Inc.

TOMORROW'S APPLES TODAY

Kevin O. Donohoe/Editor

NOTE: These products are presumed to be state of the art. Call -A.P.P.L.E. does not necessarily endorse products published in this column as **we** have not tested them.

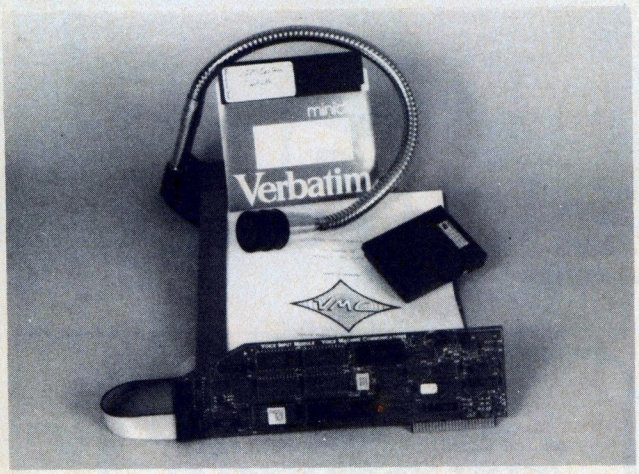
EDITOR'S NOTES:

In last month's Tomorrow's Apples Today *BASIC EXTENSION,* we left out an address and phone # for further information. It is: Foxware Products, 165 West Mead Avenue, Salt Lake City, Utah (801) 364-0394.

Complaints have risen concerning an earlier published product: SOS CLOCK /// by Systems Fabricators. Complaints of non-shipment, only recorded messages, and previously cashed checks have been filed against this product. Call —A.P.P.L.E. can only caution readers to avoid this System Fabricator product until these concerns have been resolved.

VOICE INPUT MODULE (VIM)

V.M.C.'s new VIM product adds the capability of Voice Input to *any* existing Apple II *application without programming* — from Visicalc to Apple Writer II — and allows voice in addition to keyboard entry. The VIM offers, for the first time in speech technology, near perfect word recognition — 98%+ for about $800 retail. The unit includes the Apple II compatible circuit board, the Voice Utility Program on diskette, a quality microphone with desk stand and foot-pedal switch, a User's manual and all necessary cables and connectors for immediate installation and operation. Unlimited vocabularies in subsets of up to 80 words or phrases are stored and recognized without use of the Apple's memory or processing power, since the VIM has its own microcomputer and memory.



V.M.C. will focus on Voice Input/Output for other microcomputers, a universal Voice Keyboard compatible with a wide variety of computers and a multifunction system for the handicapped. Low cost voice actuated wheelchairs, dialers, BSR controllers, electro-mechanical systems and games are also envisioned for near term development. V.M.C. is dedicated to and anticipates leading Voice Technology breakthroughs, within the next 5 years, providing a broad spectrum of applications. For further information contact: Ron Runge (714) 639-6150, 10522 Covington Circle, Villa Park, CA 92667.

HALFWAY HOUSE

The Halfway House, which bills itself as the "refuge for software addicts," is a discount software mail order house. It was the first distributor to set up a club, called the "First Look Collection," which lets members preview software for the APPLE. They pay a one-time fee to join and a small fee for each piece of software they examine.

For more information about the "First Look Collection" write to HALFWAY HOUSE, P.O. Box 22305, Lexington, KY 40522 or phone (606) 268-8703.

THE APPLE-IBM CONNECTION

The Apple-IBM Connection from Alpha Software Corp. is said to enable users to transfer information within minutes between the Apple II and the IBM Personal Computer. The package duplicates files created on one machine and transfers them into files that can be used on the other machine.

Applications for the program include allowing companies that have been developing VisiCalc models on Apple II computers to transfer these same models to VisiCalc running on the PC — without manually re-entering the data. Transfer of information developed under WordStar is also possible.

In addition to transfering files, the package is said to be the first that allows the sending and receiving of electronic mail messages between the Apple II and IBM PC.

The Apple-IBM Connection targets non-technical users by employing simple English commands and straightforward screen menus, the vendor said. Complete documentation written for professionals is also included.

The package comes with two floppy disks, one for each machine. Once the connection is made, either through a direct cable or telephone modem, a five-minute process is said to put the program into action.

Files are transferred at up to 100 characters per second.

Extensive error-checking procedures are also incorporated, the vendor said.

The Apple-IBM Connection program is priced at $195.

Alpha Software Corp., Burlington, Mass. (617) 229-2924.

68000 MACRO CROSS ASSEMBLER FOR APPLE II

S-C Software Corporation announces a new addition to their growing series of Cross Assemblers for the Apple II computer. Now Apple users can add the ability to develop programs for the Motorola MC68000 16-bit microprocessor!

The 68000 Macro Cross Assembler is a complete macro assembler with co-resident program editor. It is written in 6502 machine language for execution in the Apple II computer. It assembles standard Motorola 68000 mnemonics, using the same assembler syntax described in the Motorola reference manual.

All S-C Macro Assemblers allow powerful macros and conditional assembly. There are 20 assembler directives and 29 commands (including and EDIT command with 15 subcommands). INCLUDE and TARGET FILE capabilities allow source programs to be as large as your disk space.

The price of the S-C Macro Assembler with the 68000 Cross Assembler package is only $130.00. Registered owners of the S-C Macro Assembler may purchase the 68000 Cross Assembler package for $50.00.

S-C Software Corporation, 2331 Gus Thomasson, Suite 125, P.O. Box 280300, Dallas, TX 75228 (214) 324-2050.

TOMORROW'S APPLES TODAY

PINBALL CONSTRUCTION SET

BudgeCo announced today that it will soon be marketing the first software toy. The "Pinball Construction Set" by Bill Budge will enable a home computer owner to construct, personalize, and play his own video pinball games. Budge is the creator of Raster Blaster, the classic video pinball game and other popular graphics programs.

According to BudgeCo, no programming knowledge is required to play with this new "toy". In fact, a player doesn't even have to go to the trouble to type commands. Nothing stands between the player and the toy and all phases can be controlled under the instantaneous direction of the player just as if he were building a pinball game with his own hands.

For more details contact: BudgeCo, Ellen Beritzhoff, 428 Pala Avenue, Piedmont, CA 94611 (415) 658-8141.

QUARTET™

VISTA COMPUTER COMPANY INC.™, has recently introduced the QUARTET™, the ultimate evolutionary stage in Apple Disk Storage. The QUARTET with VISTA'S Controller & Software offers you the capacity of 4 (four) Aple Drives in the space of one at an exceptionally low cost. QUARTET'S low profile styling, utilizing 2 (two) Thinline (half high) disk drives has been developed to blend in with your Apple II and II plus Computers.

A complete QUARTET Package includes 2 (two) thinline drives that will perform in dual side/40 track and/or single side/35 track modes, controller and software patches for DOS, CP/M, Pascal.

QUARTET as described above as a Sugg. Mfgr. list of $699.00, including VISTA'S 120 Day Warranty.

For more information contact Vista Computer Company Inc. Dealer and Distributor Inquiries Invited. Local 1-714-953-0523. TELEX 910-595-1565.

THE ARCADE BOARD

The ARCADE BOARD is an amazing peripheral for the Apple II computer that generates true arcade-quality color graphics, sound effects, and music. This is because the ARCADE BOARD uses the Texas Instruments TMS 9918A Video Display Processor and the General Instruments AY3-8912 Programmable Sound Generator — the same chips that are used in many of the coin- operated video arcade machines. Best of all, the ARCADE BOARD does not require machine language to create spectacular color graphics with rapid and smooth animation and SIMULTANEOUS sound effects and music! Special hardware makes it possible to program the ARCADE BOARD entirely from Applesoft or Integer BASIC!!! In fact, the graphics and sound effects possible using only BASIC with the ARCADE BOARD far surpass any that the current crop of so-called 'arcade-quality' video games for the Apple II (all written in machine language) are capable of!! Just imagine what you can do if you chose to program the ARCADE BOARD in 6502 machine language!

The ARCADE BOARD will be available by December 1982 or January 1983. For more information or dealership inquiries write to: Millenium 3 Engineering, 1015 Gayley Ave., Suite 394, Westwood, CA 90024.

a "HOW TO" guide for users

HARDCORE COMPUTING

About DOS
- How To Write Your Own Data Base
- Using Text Files And EXEC
- Calling DOS Routines from BASIC
- Source Listings of Special DOS Routines

Programming Tips
- Peeks And Pokes
- Applesoft Tricks and Treats
- Ampersand Utilities
- Subroutine Master Library

PLUS: Lots of complete program listings from the HARDCORE program library of copyable software.

NAME _____

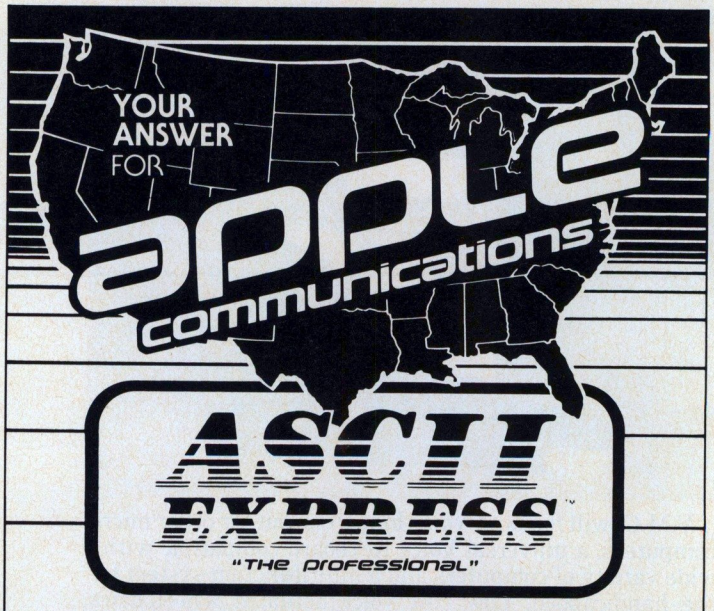
ADDRESS _____

CITY _____

STATE _____ ZIP _____

Dept. 2
P.O. Box 44549
Tacoma, WA 98444
(206) 581-6038

$20 U.S.A.
$29 Canada $33 Mexico $42 others
Sample copy: U.S.A. $5 elsewhere $8
U.S. Funds Only No Credit Cards



"THE ULTIMATE TERMINAL PACKAGE"
FOR STANDARD APPLE DOS!

Send **ANY** size or type of file: TEXT (even Random!), BINARY, APPLESOFT and INTEGER with complete error checking. • Extremely enhanced terminal emulation system supporting baud rates up to 9600 baud! • Up to 26 macro libraries — each containing complete operational parameters, along with the usual phone numbers and macro strip!

AFFORDABLY PRICED AT **$129.95**

southwestern data systems™
P.O. BOX 582-C • SANTEE, CA 92071 • 619/562-3221

CONSULTANTS

Bob Huelsdonk	Apple ///(Western U.S.)	(206) 362-4910
Stuart Yusem	Apple /// (Eastern U.S.)	(609) 822-9306
Erv Edge	Assembly (Alaska)	(907)337-7620
Dennis Baumgarte	Assembly (Eastern U.S.)	(914) 473-3210
Don Williams	Assembly (Western U.S.)	(206) 242-6807
Michael Thyng	Business	(206) 524-2744
Steven E. Bair	Data Bases	(913) 782-5901
William P. Steinberg	DOS (Eastern U.S.)	(212) 921-0663
Art Schumer	DOS (Northern U.S.)	(701) 282-7907
Lee Meador	DOS (Southern U.S.)	(817) 469-6019
Kurt Thams	DOS/Gen'l (Western U.S.)	(714) 870-9316
WM. Reynolds III	DOS/Tech (Western U.S.)	(503) 472-4028
Ted Perry	Education	(916) 485-1690
Murray Arnow	Engineering Applications	(312) 291-9210
Cornelis Bongers	European Consultant	010-525511 Ext. 3275
Alan Robinson	Fortran	(503) 757-8234
Jim Eatherly	Games	(202) 232-6046
Stan Axelrod	Hardware Design/I-O Control	(415) 488-0915
Tom Deal, M.D.	Handicapped Applications	(206) 634-2847
Randy Halvorson	Hardware Interfacing	(206) 783-9040
Ed So	Hi-Res	(408) 267-3448
Stowe Keller	Mathematics	(804) 973-5701
Fred Merchant, M.D.	Medical Group	(415) 921-7792
Stephen F. Lee	Music	(704) 289-4102
Tom Woteki	Pascal (Eastern U.S.)	(202) 547-0984
Randy Warrens	Pilot Language	(608) 837-9566
Hersch Pilloff	Stock Market	(301) 292-3100
Mark Robbins	Telecommunications	(303) 755-6440
Jim Hockenhull	Video Interfacing	(509) 334-4137
George L. Hefter	VisiCalc (East)	(617) 256-8148
Jerry Brieger	VisiCalc (West)	(206) 881-1241
Vince Aderente	VisiCalc (North)	(312) 289-9685
Dr. Bob Richard	VisiCalc (South)	(504) 466-0894
Brian Abrahams	VisiCalc (South Africa)	(Durban) 315442

AD INDEX

A.P.P.L.E.	32, 40, 55, 80
Anthro-Digital, Inc.	56
Apple Computer Inc.	74, 75
Beagle Brothers	16
Bytes & Pieces	15
Cases, Inc.	44
Datamost	4, IBC
David-Data	7
Deltrex	8
Diversified Software Research	11
Expotek	51
Futra	36
Gourmet Software	35
High Order Micro Electronics	11
IJG, Inc.	76
Ivers Specialties	53
Lama Publications	68
Lang Electronics	35
Leading Edge Products	IFC, BC
Mind Systems Corp.	49
Micro-Ware Distributing	15
Omega Microware	73
Orange Micro	12, 28
Pirates Harbor	52
Professional Data Systems	45
R.H. Electronics	6, 52
Soft Key Publishing	68
Software Management Group	27
Soph Key	78
Southwestern Data	78
Spider Software	48
Star Logic	68
Stellation Two	15, 26
Telephone Software Connection	55
Transtar	46
Vista Computer Co.	1
XPS, Inc.	38

A.P.P.L.E. ORDER FORM

DATE _____

MEMBERSHIP NUMBER _____ EXPIRATION DATE _____

NAME _____ ADDRESS _____

CITY _____ STATE _____ ZIP _____ PHONE _____

PRODUCT	UNIT PRICE	*FOREIGN SHIPPING CHARGES (See below)	AMOUNT

FOREIGN SHIPPING/HANDLING CHARGES CHART

In an attempt to simplify things for our members outside the United States, we have devised this easy rate chart for figuring the shipping and handling charges on your order. Simply determine the **TOTAL** amount of your order and add the correct amount as corresponds to the rate chart. (NOTE: THIS DOES NOT APPLY TO APPLICATIONS OR RENEWALS — BUT TO PRODUCT ORDERS ONLY!)

total amount of order	shipping/handling
$ 0.00 - $ 20.00	add $ 6.00 to order
$ 20.01 - $ 60.00	add $12.00 to order
$ 60.01 - $100.00	add $18.00 to order
$100.01 - $140.00	add $24.00 to order
$140.01 - $175.00	add $30.00 to order
$175.01 - and over	add $36.00 to order

Note to Foreign Members: All hardware products, except AP.LCASE lower case adapters, must include a United States address. These products cannot be shipped outside U.S.

VISA Account No. _____ Exp. Date _____

MasterCard Account No. _____ Exp. Date _____

Subtotal _____

**Washington sales tax _____

TOTAL AMOUNT _____

* Foreign Members Only

** Washington state residents add 6.5% state sales tax.

SEND TO: A.P.P.L.E.
304 Main Ave. S.
Renton, WA 98055

***NOTE: Canadian shipping/handling chart is being revised.**

REQUEST FOR CHANGE OF ADDRESS

Dear A.P.P.L.E.,

I am moving and my present address is:

PRESENT:

NAME _____

ADDRESS _____

CITY _____

STATE _____ ZIP CODE _____

EFFECTIVE DATE _____

NEW ADDRESS:

NAME _____

ADDRESS _____

CITY _____

STATE _____ ZIP CODE _____

MEMBERSHIP NUMBER _____

*Please fill out and return this notice two weeks

UNCLASSIFIEDS — A.P.P.L.E. ORDERS

CALL -A.P.P.L.E. (un)Classifieds

Members **Non-commerical** rate .05/word
Members commerical rate .25/word
All others: 1.00/word, $25 minimum
Send ad copy with payment to editorial office

APPLE SOFTWARE RENTAL & sales. Over 2500 programs available. Rent that expensive software for just pennies a day! Digital Delli Software. 2961 S. Las Palmas, Mesa, AZ 85202 (602) 820-0082.

ANNOUNCING ROADSEARCH for your trips. This powerful program computes the shortest practical auto routes between cities in the USA/Canada. . .and more! Database contains 406 cities and 69,000 road miles. Outputs include route, mileage, time and gallons. Printer optional. Ten day moneyback GUARANTEE. Only $34.95 (plus $1.50 s/h). Check/VISA/MC. Columbia Software, Box 2235K, Columbia, MD 21045. (301) 997-3100.

APPLE TEXT FILE TUTORIAL: Step-by-step disk-based beginner's introduction to Apple sequential text files. Teaches you to read, write, use & EXEC. DOS 3.3, 48K. $15.00 (Calif. add 6%). Send check to Computer Explanations, 2438 La Condesa Dr., Los Angeles, CA 90049.

APPLE PASCAL ANIMATION GRAPHICS. Morwe/Graphics - 83 pp. manual and two diskettes. SASE for details, $3.95 for manual, **-$19.95-** for complete kit. HENDERSON ASSOCIATES, 980 Henderson Ave., Sunnyvale, CA 94086.

CUSTOM APPLICATION programming Apple II. Fast, professional. Guaranteed, supported. 15 years experience. (206) 878-7794.

80-COLUMN PRINTER, Apple interface. Little used. $225.00. (206) 878-7794.

FOR SALE — CMI 5205 5¼" Hard Disk. Works with XEBEC Controller. Drive w/o controller. $400.00. Steve — (408) 270-2434.

PLOTTER — Watanabe Intelligent Plotter WX4671 W/interface for Apple][, 360 x 260 mm plot area 0.1 mm step size, plus paper and demo programs on disc. Excellent condition. $1100.00 (602) 829-0732.

USED SOFTWARE IN PERFECT CONDITION: Hi-Res Secrets-$50; Lisa 2.5-$40; Hi-Res Adventure #2, Pulsar II, Pig Pen, Ultima-$15 each. Phone: (617) 398-3256.

SUP-R-TERM 80 column video card, Pascal compatible, dumb terminal firmware. Like new with documentation. Cliff Niessen, 824 High St. Marquette, MI 49855 (906) 226-7864.

MUST SELL APPLE II EQUIPMENT WORKING: 48K Motherboard $450. Integer or Applesoft ROM Board $110. Communications Board $110. Disk II Analog Board $110. Add on SA400 drive for Apple $325. SA400 analog board converted for Apple use $65. 16K RAM Board $95. Replacement Keyboard $150. Drive case $39. Integer ROM set with aid #1 $35. Applesoft ROM set with autostart $45. MC multifunction card with software/manual $140. Eurapple II motherboard, no TTL, RAM, ROM $225. Much, Much, More! NON-WORKING: Mountain Computer (MC) Music System (boards only) $95. MC A/D-D/A converter $95. MC bare expansion board $95. NON-WORKING NO-TTL: Apple II motherboard $250. Serial, Parallel, Communication, Language, ROM, or Disk analog board $45. ANY THREE $120. ALL SIX $200. NO REASONABLE OFFER REFUSED. RICHARD SMITH, 2961 S. Las Palmas, Mesa, AZ 85202 (602) 820-0082.

SEND FOR YOUR FREE CATALOG FROM TECK ASSOCIATES — producers of top quality computer assisted instructional materials in BASIC SKILLS, AUTHORING, SCIENCE and EARLY LEARNING. Contact TECK Associates, Box 8732, Dept. CA, White Bear Lake, MN 55110.

BOOK KEEPER II with CHECK WRITER II. Yes, a unique single entry BookKeeping system for the Apple. As your write your checks, maintain up to 300 General Ledger Accounts, and unlimited Account distribution. Excellent for BUSINESS or PERSONAL use. Have 300 Payable vendors on file. Have 300 Ledger Acct. Numbers on file. Maintain Monthly and YTD Totals. MARK/DONN ASSOC., BOX 1589, MARIETTA, GA 30061 (404-422-8169). Visa or Master Charge. Dealer inquiries invited.

MOUNTAIN CPS BOARD. Hardly used, with CP/M, Applesoft and Pascal software. $140. (206) 545-1005 — daytime; (206) 546-2193 — evenings.

SILENTYPE PRINTER — with interface, manual and 1 roll paper. $200.00. S. Bohn, 6822 Emerald Ave., Enon, Ohio 45323, (513) 864-2754.

PRE-OWNED SOFTWARE CO.—Sell your pre-owned software on consignment. Rent pre-owned software $5 week min. Buy pre-owned software at 50% discount. Call (206) 784-2081 or send SASE P.S. 4229 Baker Ave. NW, Seattle, WA 98107 for details.

CLOSE OUT SALE—4 Lazer lower case + @ $40.00 ea and 6 keyboard + @ $60.00 ea. David Hansmeyer, 7323 Glen Haven, San Antonio Tx 78239.

STELLAR SPELLER: Help your child learn his spelling words at home. FLEXIBLE: allows you to choose the words that your child will practice. Permits changes to existing lists, creation of new lists. EFFECTIVE: Provides challenge and reinforcement to make learning fun. For 48K Apple II plus with DOS 3.3. Price $40 for system, $10 for manual only. COTTAGE INDUSTRIES, 112 Locksley, Dept CA, San Francisco, CA 94122.

FOR SALE: MICROSOFT RAM CARD, unopened, $80 or best offer. Scott Joslyn, 3929 E. 5th St., Long Beach CA 90814, (213) 438-2628 EVE.

AMWAY PROGRAMS for distribution — SA1 and bonus. Frank Luth, Jr., Box 519, Port Republic, VA 24471.

Now Available!

All About Pascal

Call —A.P.P.L.E. IN DEPTH II

184 page reference tool containing a wide range of new and useful articles/programs specializing in Apple Pascal.

Member Price:

$11.00

Non-member price: $12.50
(Available at your local computer store)

A.P.P.L.E. Orders
304 Main Ave. S.
Renton, WA 98055

Washington state residents add 7.0% sales tax
Foreign members add $6.00 for shipping/handling charge
Canadian members, postage being revised

Dealer inquires invited

PRODUCTS NOW AVAILABLE TO A.P.P.L.E. MEMBERS

Call —A.P.P.L.E. Specials

Word Weaver Special[1]	$60.00
ApGraphics: The Filler Introductory Price[2]	$17.00

A.P.P.L.E. LIBRARY

A.P. Drive Technical Manual	$30.00
"Apple Orchard" No. 1 (magazine)	$2.50
Apple Users Guide	$13.00
Assembly Lines: The Book	$16.50
Beneath the Apple DOS	$17.00
"Best of Cider Press"	
(S.F. Newsletter Annual) Vol I	$4.50
(S.F. Newsletter Annual) Vol II	$9.50
Call —A.P.P.L.E. annuals	
1978	$7.50
1979	$10.00
1980	$15.00
1981	$18.50
Call —A.P.P.L.E. In Depth No.1	$7.00
Call —A.P.P.L.E. In Depth No. 2	$11.00
Kids and the Apple	$16.00
Power of VisiCalc Vol I or II, each	$8.50
Using 6502 Assembly Language	$17.00
Woz Talks (VHS Video cassette)	$27.50

AS IS SOFTWARE

Public domain diskettes (DOS 3.2.1) (Send 37¢ S.A.S.E. for list of titles)	ea. $4.00

A.P.P.L.E. DISKPAKS

Call —A.P.P.L.E. Anthology Vols I-VII, incl.	ea. $12.50
DICE 82[3]	$12.50
Diskpak 12	$16.50
Diskpak 34A	$13.00
Diskpak 5	$13.00
Diskpak 6B (DOS 3.2 disk utility)	$18.50
Diskpak 6B-16 (DOS 3.3 disk utility)	$18.50
Diskpak 7	$17.50
Diskpak 8	$13.00
In Depth 1 Diskette	$16.50
In Depth 2 Diskette Set	$20.00
Pascal Anthology '81	$12.50

A.P.P.L.E. HARDWARE

ApController	$95.00
Ap.Coolkit	$22.50
AP.DRIVE	$325.00
AP.DRIVE with Controller	$420.00
Ap L Case 1	$30.00
Ap L Case 2 (Lower case chip)	$27.50
Disk Banks	2 for $9.00

AP•Jewelry

Lapel pin	$3.00

Apple Prestige Product LinE

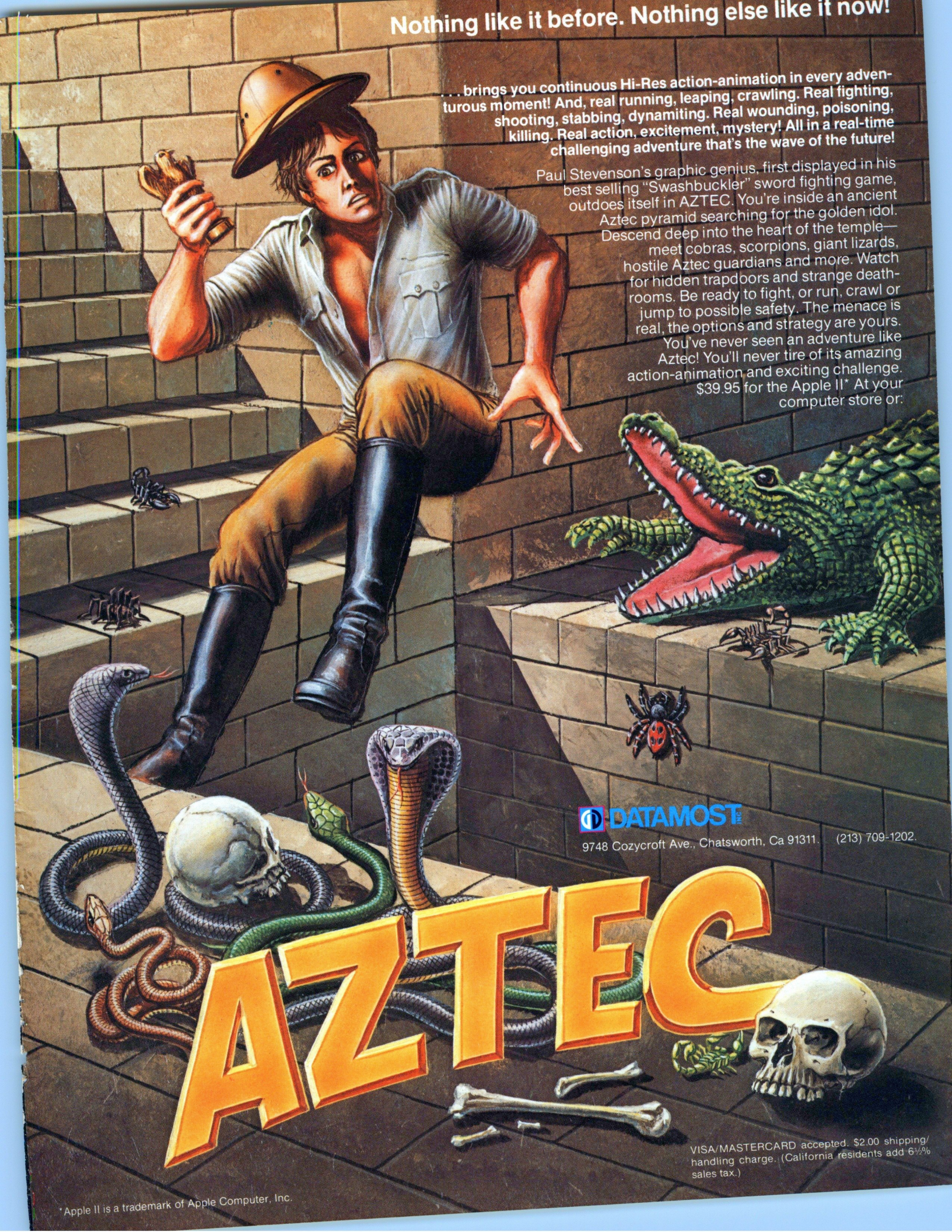
APfile — Text file utility	$18.50
ApGraphics: The Filler	$22.00
Applesoft Carpenter	$18.50
Aptest	$18.50
Big Mac Macro-Assembler/Ted	$22.50
Big Mac.LC	$28.50
Big Mac X-Ref	$7.50
Global Program Line Editor	$38.50
Higher Fonts 1	$13.00
Higher Text Plus	$27.50
Program Global Editor	$18.50
Program Line Editor	$20.00
Soft Seventy	$22.00
Symbol Symon	$20.00
WordWeaver ///	$75.00

[1]Expires January 31, 1983
[2]Expires March 1, 1983
[3]$7.50 With Ap L Case 1
$10.00 With Ap L Case 2

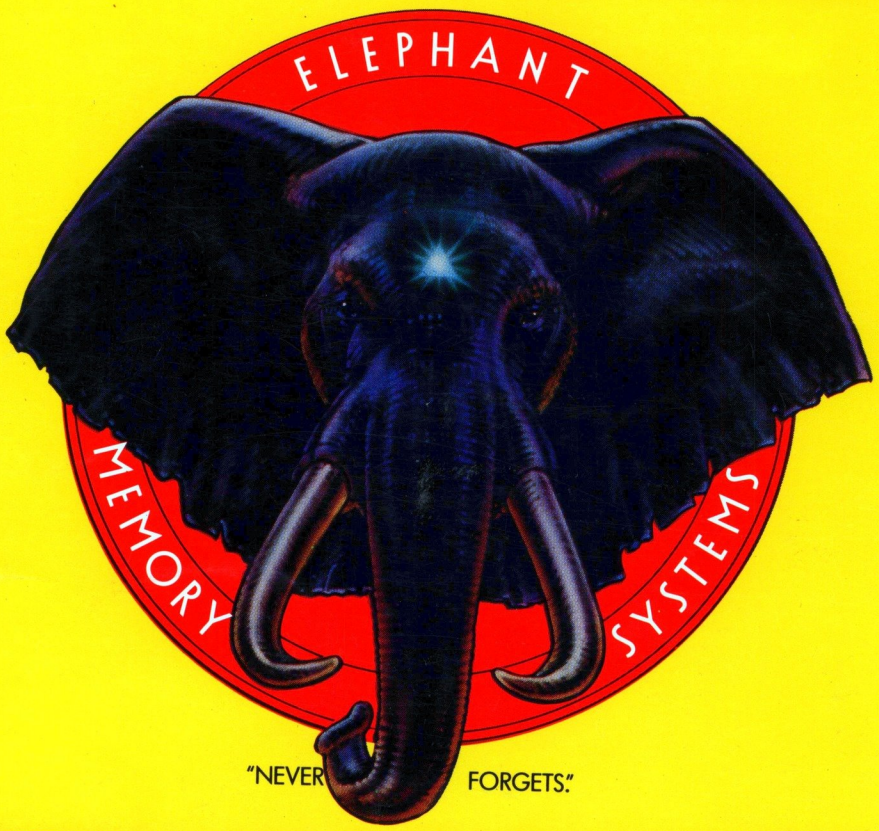
Diskettes supplied on DOS 3.3 except as noted.
All walkin orders must be called in prior to pickup.

NOTICE TO FOREIGN MEMBERS:
A.P.P.L.E. Hardware products not available to overseas or foreign members. This includes Ap.Drive & Ap Cool Kit.

ORDER FORM ON REVERSE


Nothing like it before. Nothing else like it now!
. . . brings you continuous Hi-Res action-animation in every adventurous moment! And, real running, leaping, crawling. Real fighting, shooting, stabbing, dynamiting. Real wounding, poisoning, killing. Real action, excitement, mystery! All in a real-time challenging adventure that's the wave of the future!
Paul Stevenson's graphic genius, first displayed in his best selling "Swashbuckler" sword fighting game, outdoes itself in AZTEC. You're inside an ancient Aztec pyramid searching for the golden idol. Descend deep into the heart of the temple—meet cobras, scorpions, giant lizards, hostile Aztec guardians and more. Watch for hidden trapdoors and strange death-rooms. Be ready to fight, or run, crawl or jump to possible safety. The menace is real, the options and strategy are yours. You've never seen an adventure like Aztec! You'll never tire of its amazing action-animation and exciting challenge. $39.95 for the Apple II* At your computer store or:
DATAMOST
9748 Cozycroft Ave., Chatsworth, Ca 91311. (213) 709-1202.
AZTEC
VISA/MASTERCARD accepted. $2.00 shipping/handling charge. (California residents add 6½% sales tax.)
*Apple II is a trademark of Apple Computer, Inc.

REMEMBER:



MORE THAN JUST ANOTHER PRETTY FACE.

Says who? Says ANSI.

Specifically, subcommittee X3B8 of the American National Standards Institute (ANSI) says so. The fact is all Elephant™ floppies meet or exceed the specs required to meet or exceed all their standards.

But just who is "subcommittee X3B8" to issue such pronouncements?

They're a group of people representing a large, well-balanced cross section of disciplines—from academia, government agencies, and the computer industry. People from places like IBM, Hewlett-Packard, 3M, Lawrence Livermore Labs, The U.S. Department of Defense, Honeywell and The Association of Computer Programmers and Analysts. In short, it's a bunch of high-caliber nitpickers whose mission, it seems, in order to make better disks for consumers, is also to make life miserable for everyone in the disk-making business.

How? By gathering together periodically (often, one suspects, under the full moon) to concoct more and more rules to increase the quality of flexible disks. Their most recent rule book runs over 20 single-spaced pages—listing, and insisting upon—hundreds upon hundreds of standards a disk must meet in order to be blessed by ANSI. (And thereby be taken seriously by people who take disks seriously.)

In fact, if you'd like a copy of this formidable document, for free, just let us know and we'll send you one. Because once you know what it takes to make an Elephant for ANSI . . .

We think you'll want us to make some Elephants for you.

ELEPHANT.™ HEAVY DUTY DISKS.

For a free poster-size portrait of our powerful pachyderm, please write us.
Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021
Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.